Classifying Injury Outcome for Basketball Players in the National Basketball Association

In this project I applied machine learning to analyze the features influencing injury outcome and predict whether or not a player would be injured in a given NBA season.

Introduction

Sports are a global source of entertainment that generate billions of dollars and attract millions of viewers each year. Teams/players are constantly trying to optimize performance and boost their chances of winning. One of the largest barriers to winning is injury. When a team's player is injured this hurts the team's chances of winning. The National Basketball Association (NBA), one of the world's most watched and profitable sports leagues, is especially impacted by this issue. Injuries to star players decrease a team's chance of winning, profit, and viewership. Beyond financial implications, injuries place significant physical and emotional strain on the athletes themselves. Understanding what factors lead to injury can help to reduce the occurrence of injuries and lessen their consequences for players, teams, and the league as a whole.

I utilized three datasets from Kaggle to identify which factors distinguish players who suffer injuries from those who remain healthy over the course of a season. One dataset contained injury reports, another provided player workload and biometric information, and the third included game-recorded statistics. My goal was to determine whether season-level aggregated player statistics could reliably classify injury occurrence. The availability of this well-labeled, quantitative data made supervised machine learning a viable potential solution. I used a random forest classification algorithm to train and test the model to evaluate which features contributed most to predicting injury. I hypothesized that higher usage, workload, age, and rebound frequency would be associated with increased injury likelihood.

I created a dataframe of aggregated season data and used this data to train a classification model. The model identified that the amount of games played in a season was the most important feature for distinguishing between players who got injured and those who remained healthy. The accuracy of the model was moderate, and below the desired threshold.

DATA

The first dataset I used included individual box scores (every statistic an individual can record during a basketball game) for every regular season NBA game from 2000–2024. To prepare this data to be merged and compared with the other datasets I needed to convert date strings into datetime objects; the column for player names also needed to be renamed for merging. I then needed to filter the years for the timeline of NBA seasons. I found that the NBA regular season runs from October to May of the following year, and I confirmed this with my data, shown in the plot below. I then filtered the ‘stats’ dataframe to include the 2000–2020 NBA season. I restricted the analysis to this time period because it offered the maximum overlap across all three datasets. To compute the ‘stats’ data for player age and for the number of minutes played I wrote two functions to convert the string objects into numerical ones to be utilized in training the model.

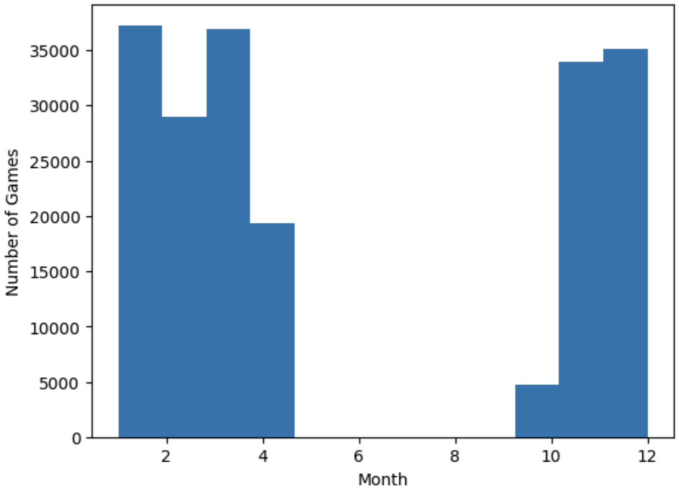

Figure 1: Number of NBA Games Per Month

The second dataset was an NBA injury report spanning from 1951–2023. To clean this ‘injury’ dataset I needed to filter it to align with the same date range as the ‘stats’ dataset and have the exact same column name for storing player names. I also removed unnecessary columns from the ‘injury’ dataset and stripped the Player Name column of extra spaces to avoid matching issues.

The last dataset I used included player heights, weights, and usage percentage for all players from 1996–2023. I called this dataset ‘anthro’. I filtered ‘anthro’ to span from 2000-2020 like ‘stats’ and ‘injury’ I then removed unnecessary columns and aggregated the features of this dataset to get the mean values for each player. The aggregation prevented extra rows being created when ‘anthro’ was merged with ‘stats’. Finally I changed the Players Name column to match both the ‘stats’ and ‘injury’ dataframes.

After preprocessing ‘anthro’ and ‘stats’, I merged these two dataframes together with a left merge using the player name column. This kept all the data from ‘stats’ and added all the data columns from ‘anthro’ to the corresponding player name. Players in the ‘anthro’ dataset that were not in the ‘stats’ dataset were lost.

With the merged ‘stats’ and ‘anthro’ dataframe and the ‘injury’ dataframe, I created a row for every player from ‘stats’ for each season with their averaged feature variables and injury classification for 20 seasons. The average value was calculated using all of the games a player recorded statistics in for that season, aside from biometric data such as height. The injury classification was set by reviewing the ‘injury’ dataset for any instances the player was injured during the season. If the player was injured during that season the ‘injury indicator’ column value is set to 1, but if the player was uninjured during the season the value is set to zero. This column was the target variable for the model. All of the rows were then appended to a final dataframe used for training and testing the model.

A caveat of the code was that every player had 20 rows of data even if they did not play 20 seasons. These fake seasons contained NaN values so I dropped the rows with NaN values, thus eliminating seasons without real data. Dropping NaN values also removed rows with missing information and ensured that all remaining rows contained a complete and consistent set of features.

The final data frame composed 8 features that were used to train the model and the class distribution for the target feature ‘injury indicator’ was almost evenly split.

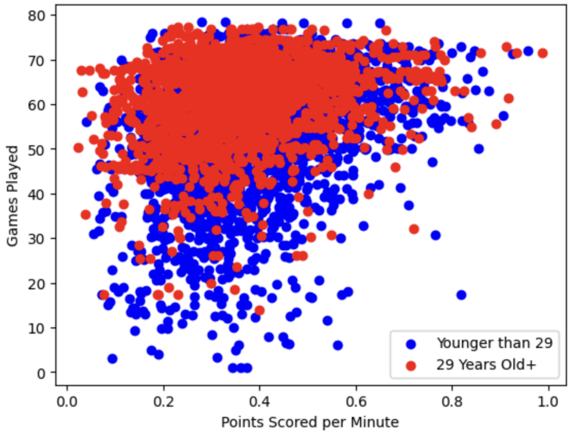

Figure 2: Points per Minute vs Games Played

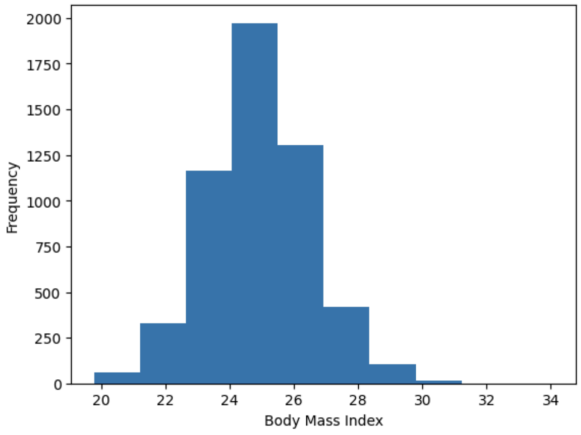

Figure 3: Histogram of BMI

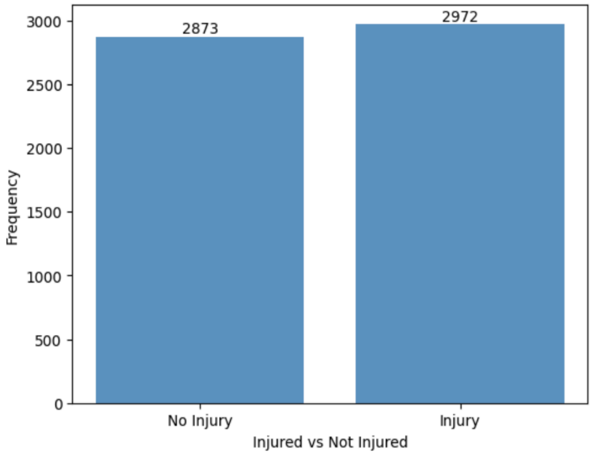

Figure 4: Histogram of Seaons with Injuries and Seaons without Injuries

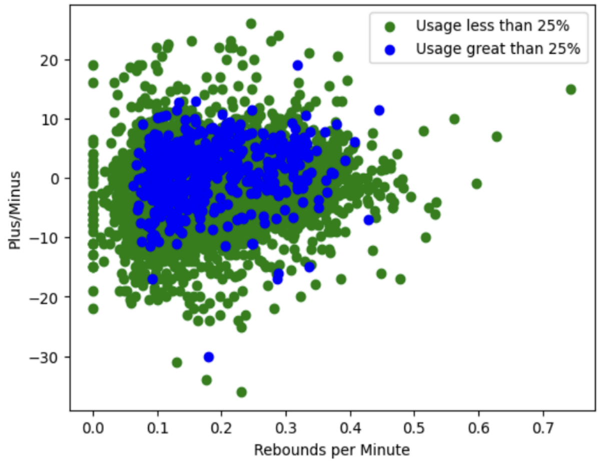

Figure 5: Rebounds per Minute vs Plus/Minus

Input features:

- Number of games played
- Age of player [in years]
- Body mass index of player
- Average minutes played per game
- The average score differential while a player was in the game (plus/minus)
- Average player usage (the percentage of a team's possessions a player uses while on the court, ending with a field goal attempt, free throw attempt, or turnover)
- Average number of rebounds recorded per minute when playing
- Average number of points scored per minute when playing

The age of each player and the number of games played were taken directly from the original datasets, while the remaining variables were engineered. Body mass index (BMI) was calculated by converting player height to meters and dividing weight (in kilograms) by height squared. Average minutes played per game, average plus/minus, and average usage were computed by taking the average across all games played in the season. Rebounds per minute and points per minute were created by dividing each player's average rebounds and points per game by their average minutes played per game. All engineered features were calculated and assigned to the corresponding player-season for every game played between 2000 and 2020.

I selected these features because each one reflects a factor that could plausibly influence injury risk in NBA players. Age captures the expected decline in athletic durability and recovery capacity with time. Games played and average minutes played provide measures of cumulative workload and duration of physical stress. Usage rate, points per minute, and rebounds per minute reflect how intensely a player is involved in offensive or defensive plays, which may increase mechanical strain and fatigue. Points per minute and rebounds per minute also measure a player's relative physical output, where larger values correspond to higher levels of exertion and activity. Plus/minus offers a proxy for overall on-court impact and playing context, while BMI incorporates biometric information that may relate to joint loading and fitness level. By combining workload, biometric, and performance-intensity metrics, this feature set was chosen to capture multiple dimensions of a player's physical demands throughout a season that could reasonably influence injury likelihood.

Model

The project was framed as a classification problem because the goal was to predict whether a player would sustain an injury in an NBA season, a discrete outcome rather than a continuous value. The Random Forest classifier from scikit-learn was chosen because it naturally handles complex, nonlinear relationships between player statistics and injury outcomes, while remaining robust to outliers and variability in the data. Since the dataset was well-labeled, a supervised learning approach was appropriate.

Random forest uses multiple estimators (decision trees) with bootstrap sampling to reduce overfitting. Plus, each tree uses randomized data samples to account for outliers. After computing each tree, the ensemble combines all trees to determine the most informative splits. Each split aims to maximize class purity, measured by Gini impurity, minimizing the likelihood that a randomly chosen observation would be misclassified. The model also provides feature importance measures, enabling interpretable insights into which statistics most influence injury risk. Random Forests are well-suited to handle my medium-sized dataset and can accommodate class imbalances. The goal of this project was to identify what distinguishes players who get injured from those who do not and the Random Tree Classifier is suited to achieve this.

The tree below is an example of one component of the Random Forest ensemble, which contributes to the model’s overall decision-making. This tree makes decisions based only on the random subset of data it is exposed to, using Gini impurity to determine the optimal splits. Its first split is based on the number of games played, indicating that separating the data by this feature most effectively reduces the likelihood of misclassification for this tree.

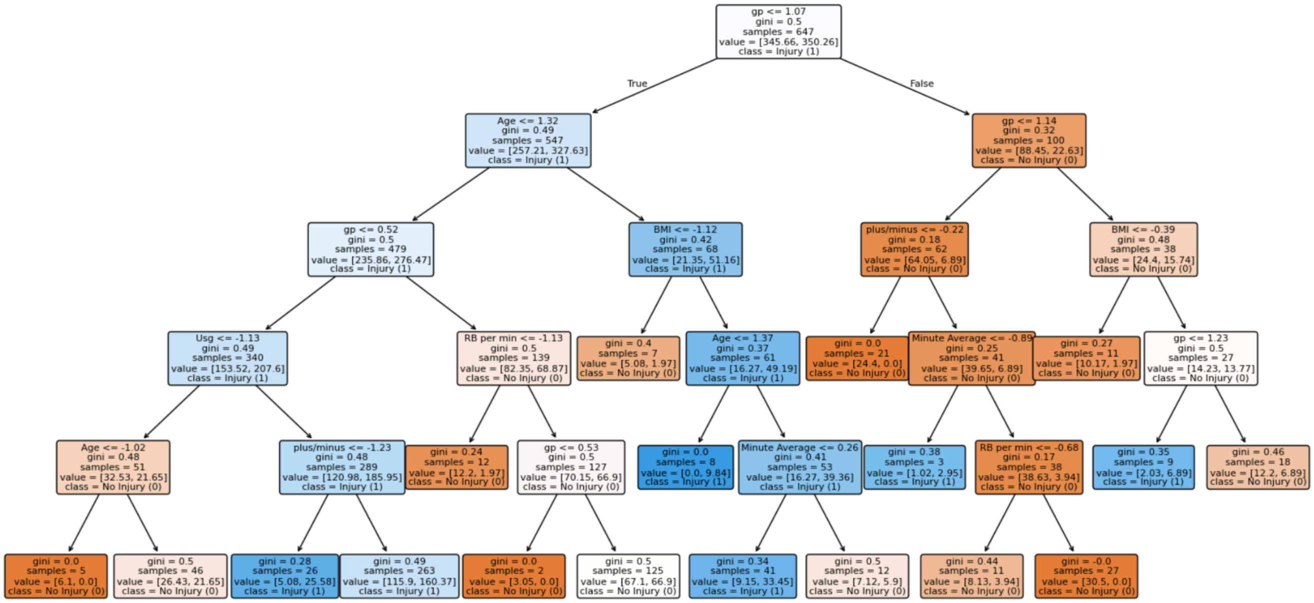
gp <= 1.07 gini = 0.5 samples = 647 value = [345.66, 350.26] class = Injury (1)
True
False
Age <= 1.32 gini = 0.49 samples = 547 value = [257.21, 327.63] class = Injury (1)
gp <= 1.14 gini = 0.32 samples = 100 value = [88.45, 22.63] class = No Injury (0)
gp <= 0.52 gini = 0.5 samples = 479 value = [235.86, 276.47] class = Injury (1)
BMI <= -1.12 gini = 0.42 samples = 68 value = [21.35, 51.16] class = Injury (1)
plus/minus <= -0.22 gini = 0.18 samples = 62 value = [64.05, 6.89] class = No Injury (0)
BMI <= -0.39 gini = 0.48 samples = 38 value = [24.4, 15.74] class = No Injury (0)
Usg <= -1.13 gini = 0.49 samples = 340 value = [153.52, 207.6] class = Injury (1)
RB per min <= -1.13 gini = 0.5 samples = 139 value = [82.35, 68.87] class = No Injury (0)
gini = 0.4 samples = 7 value = [5.08, 1.97] class = No Injury (0)
Age <= 1.37 gini = 0.37 samples = 61 value = [16.27, 49.19] class = Injury (1)
gini = 0.0 samples = 21 value = [24.4, 0.0] class = No Injury (0)
Minute Average <= -0.89 gini = 0.25 samples = 41 value = [39.65, 6.89] class = No Injury (0)
gini = 0.27 samples = 11 value = [10.17, 1.97] class = No Injury (0)
gp <= 1.23 gini = 0.5 samples = 27 value = [14.23, 13.77] class = No Injury (0)
Age <= -1.02 gini = 0.48 samples = 51 value = [32.53, 21.65] class = No Injury (0)
plus/minus <= -1.23 gini = 0.48 samples = 289 value = [120.98, 185.95] class = Injury (1)
gini = 0.24 samples = 12 value = [12.2, 1.97] class = No Injury (0)
gp <= 0.53 gini = 0.5 samples = 127 value = [70.15, 66.9] class = No Injury (0)
gini = 0.0 samples = 8 value = [0.0, 9.84] class = Injury (1)
Minute Average <= 0.26 gini = 0.41 samples = 53 value = [16.27, 39.36] class = Injury (1)
gini = 0.38 samples = 3 value = [1.02, 2.95] class = Injury (1)
RB per min <= -0.68 gini = 0.17 samples = 38 value = [38.63, 3.94] class = No Injury (0)
gini = 0.35 samples = 9 value = [2.03, 6.89] class = Injury (1)
gini = 0.46 samples = 18 value = [12.2, 6.89] class = No Injury (0)
gini = 0.0 samples = 5 value = [6.1, 0.0] class = No Injury (0)
gini = 0.5 samples = 46 value = [26.43, 21.65] class = No Injury (0)
gini = 0.28 samples = 26 value = [5.08, 25.58] class = Injury (1)
gini = 0.49 samples = 263 value = [115.9, 160.37] class = Injury (1)
gini = 0.0 samples = 2 value = [3.05, 0.0] class = No Injury (0)
gini = 0.5 samples = 125 value = [67.1, 66.9] class = No Injury (0)
gini = 0.34 samples = 41 value = [9.15, 33.45] class = Injury (1)
gini = 0.5 samples = 12 value = [7.12, 5.9] class = No Injury (0)
gini = 0.44 samples = 11 value = [8.13, 3.94] class = No Injury (0)
gini = -0.0 samples = 27 value = [30.5, 0.0] class = No Injury (0)

Figure 6: A single decision tree from the Random Forest, illustrating that the first split consistently occurs on Games Played

The model was developed by importing the necessary components, specifying the target feature and input features, splitting the data into train and test sets, building the pipeline with the ideal parameters, fitting the model, and testing the model. The code is shown below.

```
#####
### input features and target feature
#####
feature_cols = ['Minute Average', 'Age', 'plus/minus', 'BMI', 'RB per min', 'Usg',
'pt_per_min', 'gp'] #'Weight', 'Height',
X = df_ml[feature_cols]
y = df_ml['Injury Severity']
#####
# scikit-learn spilt train and test data
#####
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
#####
# make pipeline for random forest
#####
pipeline = Pipeline([("scaler", StandardScaler()),("rf",
RandomForestClassifier(n_estimators=200, max_samples=0.2, bootstrap=True,
class_weight='balanced',max_depth=4, min_samples_split = 15))])
#####
# fit model
#####
pipeline.fit(X_train, y_train)
#####
# perform cross validation
#####
cv_scores = cross_val_score(pipeline, X, y, cv=10)
#####
# test model
#####
```

```
y_pred = pipeline.predict(X_test)
```

Results

Below are the individual results for a 10 fold cross validation and the mean accuracy across all folds. This value shows that the model predicts whether a player will be injured correctly about 60.5% of the time on average.

Cross-validation accuracy for each fold: [0.65, 0.63103448, 0.63448276, 0.54310345, 0.58275862, 0.59827586, 0.60344828, 0.61206897, 0.55958549, 0.64075993]

Mean CV accuracy: 0.6055517836936455

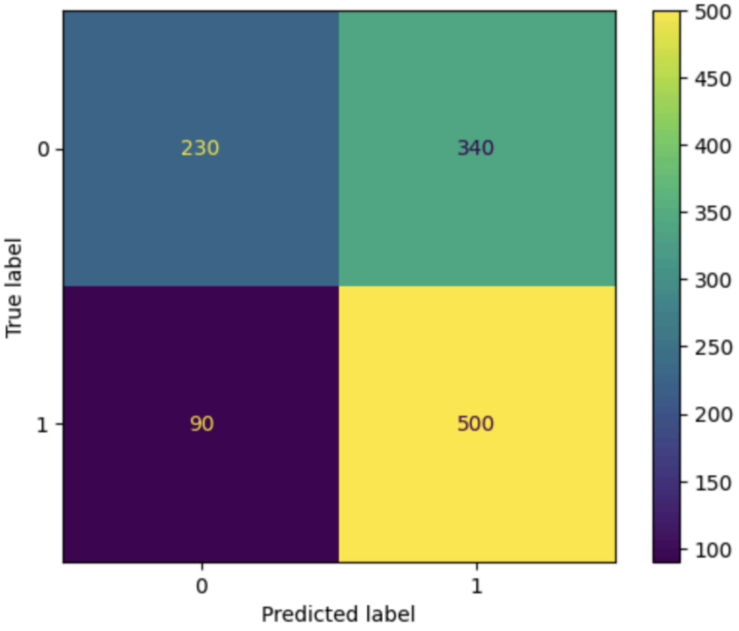

Figure 7: Results from the confusion matrix, displaying the predictions from model testing

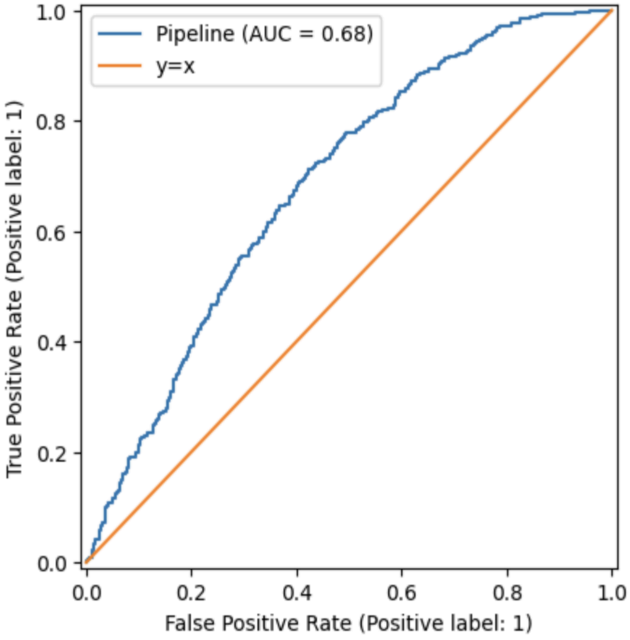

Figure 8: ROC curve for the model, displaying the model's relative success at separating the injured and non-injured player classes

The confusion matrix (Figure 7) shows that the model performs better at predicting players who are not injured, achieving an accuracy of approximately 71%. Its performance is lower for predicting injured players, around 60%, and it produces a notable number of false positives. The ROC curve (Figure 8) indicates that the model performs better than random chance, but overall accuracy remains modest, highlighting potential issues with the data or model.

The feature importance for the model is as follows:

1. Games Played–0.243297
2. Usage–0.176144
3. BMI–0.165704
4. Minute Average–0.121725
5. Points Per minute–0.080330
6. Rebounds Per minute–0.072050
7. Age–0.071117
8. Plus/Minus–0.069633

The feature importance value represents the contribution a feature made to improve the purity of the ensemble overall. High values indicate more impactful features.

The 8 partial dependence plots below each show how the average predicted outcome for the model changes based on the value of each feature, independent of other features. Y-axis values closer to 1 indicate the average prediction is closer to (1), meaning the model believes the player will suffer an injury; whereas values closer to (0) indicate the average prediction is more likely to be that a player will not suffer an injury.

The PDPs show greater values of points per minute (Figure 15), usage (Figure 14), rebounds per minute (Figure 13), and age (Figure 10) contribute to the model predicting the player will be injured (1). Conversely, the PDPs for games played (Figure 16) and average minutes played (Figure 9) show that greater values contribute to the model predicting no injury (0). The PDPs for BMI and Plus/Minus (Figures 11 & 12) do not show a strong trend toward predicting either class.

Partial Dependence Plots (PDPs) for Each Input Feature

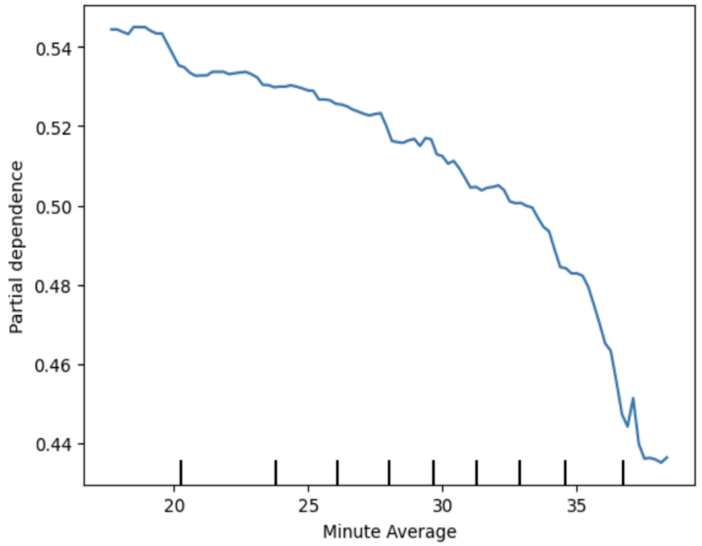

Figure 9: Partial Dependence Plot for Average Minutes Played

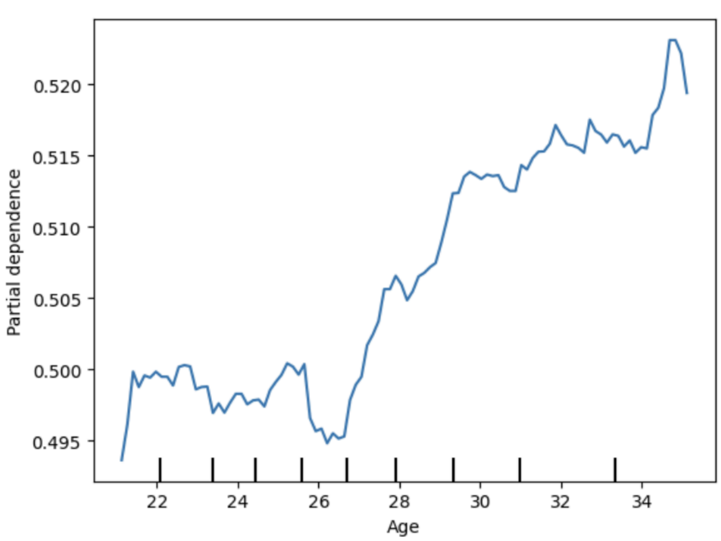

Figure 10: Partial Dependence Plot for Average Age

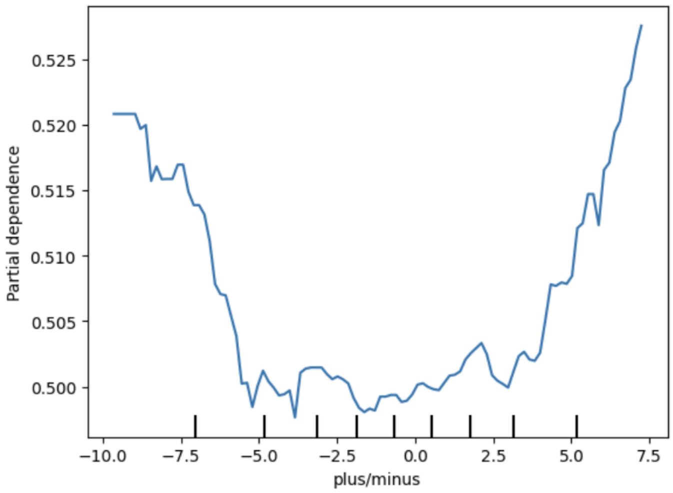

Figure11: Partial Dependence Plot for Plus/Minus

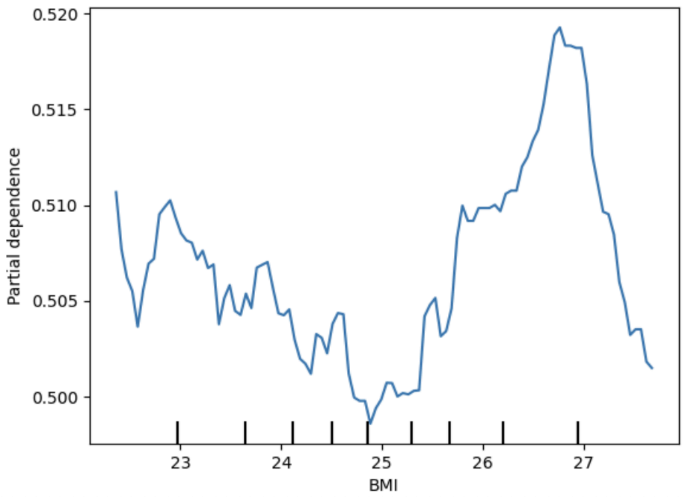

Figure 12: Partial Dependence Plot for Body Mass Index

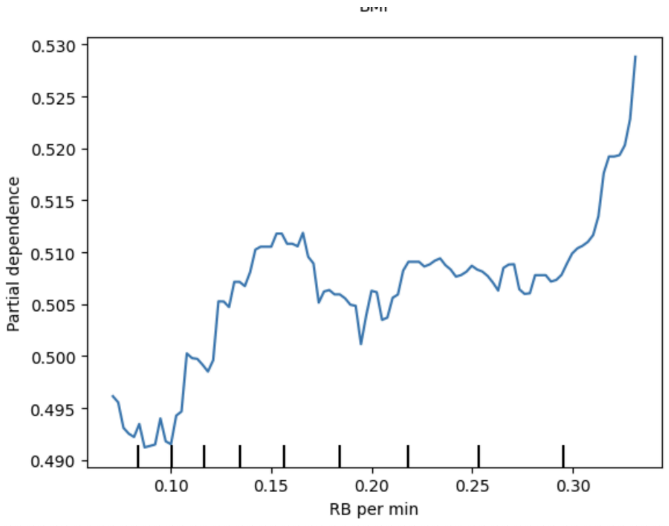

Figure 13: Partial Dependence Plot for Average Rebounds Recorded per Minute

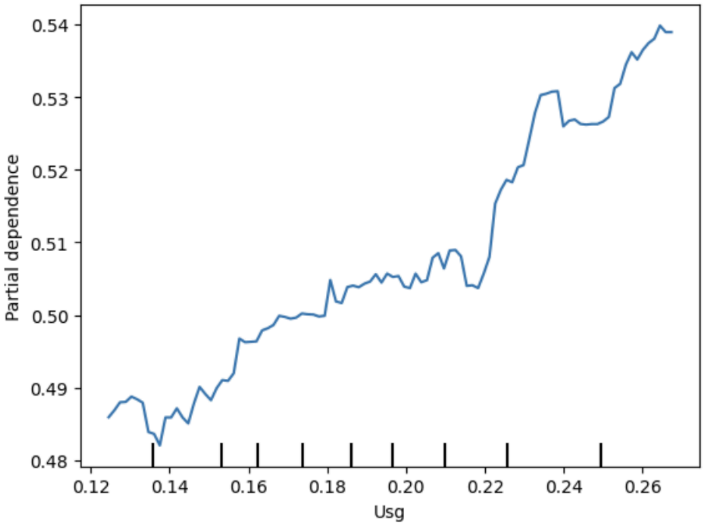

Figure 14: Partial Dependence Plot for Average Usage

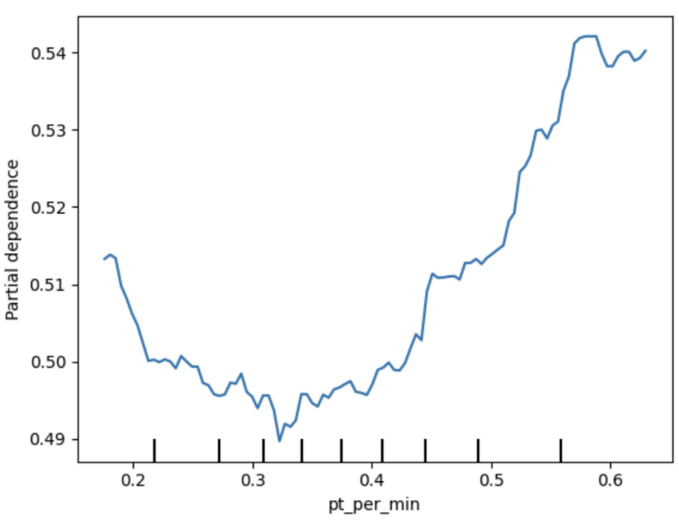

Figure 15: Partial Dependence Plot for Average Points Scored per Minute

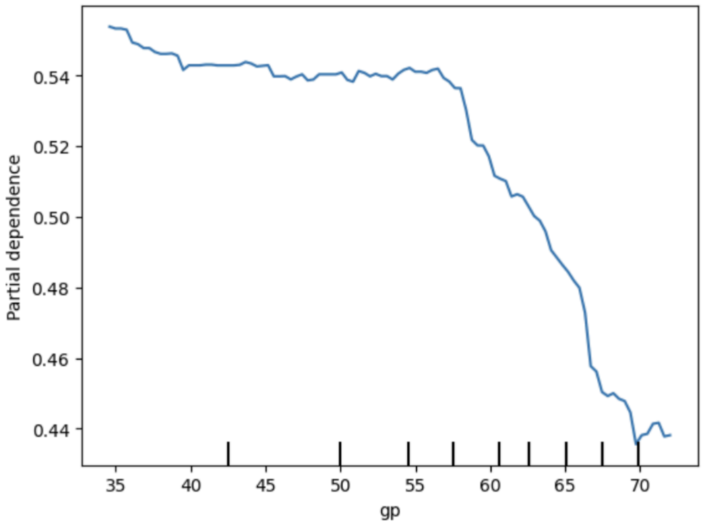

Figure 16: Partial Dependence Plot for Average Games Played

Discussion

The moderate performance of the model, reflected in both the mean cross-validated accuracy and the confusion matrix, indicates clear limitations in its ability to predict injury occurrence. To determine whether these shortcomings were due to the Random Forest Classifier itself, I evaluated several alternative algorithms, including an Extra Trees Classifier, Gradient Boosting Classifier, AdaBoost Classifier, and a Support Vector Machine. For each classifier I tuned the parameters—max depth, class weights, max samples, etc.—to attempt to improve the model, but none produced improvements. The mean cross-validated accuracies of these models remained between 55% and 62%, and the AdaBoost model frequently collapsed to predicting the injury class for nearly all samples. These results suggest that the choice of classifier was not the main constraint.

Instead, the more likely cause of the model's performance is the structure and quality of the input data. The features I gathered—along with the additional engineered variables—appear insufficient to reliably distinguish between injured and non-injured player-seasons. In particular, season aggregated statistics might obscure meaningful short-term variations or patterns related to injury risk. Although the input features spanned a broad range of categories—including play style, team contribution, game impact, body characteristics, and overall performance—there was substantial overlap between injured and non-injured players across nearly all variables. This overlap suggests that the chosen features do not adequately separate the two groups, a conclusion further supported by the partial dependence plots. The plots show that none of the features exhibit a very strong effect on the model's predictions, as the predicted probabilities remain close to 0.5 across all features indicating weak class separation.

Even though the model determined that the number of games played was the most important feature, this is not reflected in the partial dependence plot. TThe plot suggests that more games played slightly increases the likelihood of predicting no injury, but the magnitude of this effect is small—the lowest average prediction hovers around 0.44, far closer to 0.5 than to either class boundary. It is unlikely that the feature importance results of this model are robust and would need to be further verified by future work to be meaningful.

Conclusion

In summary, this project did not produce a model capable of accurately classifying players who get injured from players who remain healthy during an NBA season. The Random Forest Classifier was likely not the cause of the moderate performance because the use of other estimators and a myriad of alternative parameters did not improve performance. The partial dependence plots display that none of the input features were strongly distinguishable based on class, and likely the cause of the model's shortcomings. Therefore, the feature-importance rankings—particularly the finding that games played appears most important—should not be treated as definitive.

Overall, the results of this project suggest that high-level, season-aggregated NBA statistics are not sufficient to meaningfully predict injury occurrence. Reliable prediction will likely require additional biomechanical and workload-specific data. The modern era of the NBA now collects a vast array of biometric and player-tracking information that was not available in earlier seasons. The difference between the data recorded in 2000 and the data available in 2025 is enormous, and this more individualized and granularized information may hold the key to accurate injury prediction. For example, current player-tracking systems record detailed metrics such as jump distance, time spent in the air, power of liftoff, verticality, movement patterns, and acceleration forces. Future studies could potentially use these newly available data features and shorter time scales to better capture the factors that influence injuries. Incorporating this type of detailed biomechanical information as new training features could substantially improve the model's ability to predict injury outcomes in future work.

References

'Anthro' Data https://www.kaggle.com/datasets/justinas/nba-players-data?select=all_seasons.csv

'Injury' Data https://www.kaggle.com/datasets/loganlauton/nba-injury-stats-1951-2023

'Stats' Data https://www.kaggle.com/datasets/bme3412/nba-player-stats-20002020-season