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. The organizations and players who compete in these competitions are all trying to win. More and more data analytics have helped teams/players to optimize performance and boost success, but a key barrier to winning is injuries. When a team's player is injured this hurts the team's chances of winning and disappoints sports fans. 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 cause fans to lose interest and stop watching games. Injuries also decrease a team's chance of winning, which can hurt ticket, merchandise, and concessions sales. Beyond financial implications, injuries place significant physical and emotional strain on the athletes themselves. Gaining a better understanding on what factors lead to injury can help to potentially reduce the occurrence and resulting consequences of injuries.

I utilized three datasets, all available on Kaggle, to attempt to identify what distinguished players who get injured from those who remain healthy during a season. 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 determine what feature contributes most to injury.

After cleaning the data and training the model with the filtered data I found 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 not as high as I hoped.

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. To

compute the ‘stats’ data for player age and for the number of minutes played I wrote two functions to convert the string object into a numerical one.

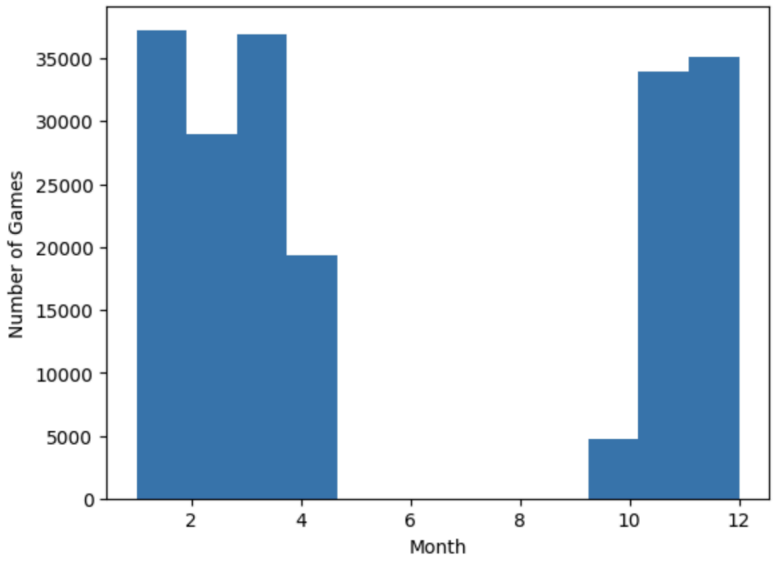

Figure 1: Number of NBA Games Per Month

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 every row of extra spaces.

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 and removed unnecessary columns. I also aggregated the important features of this dataset to get the mean values for each player. Finally I changed the players name column to match both the ‘stats’ and ‘injury’ dataframes.

After cleaning ‘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 from ‘anthro’ with a corresponding player name. Players in the ‘anthro’ dataset that were not in the ‘stats’ dataset were lost. This action added four new features for each player’s games for all 20 NBA seasons under analysis.

Using merged the ‘stats’ and ‘anthro’ dataframe with the ‘injury’ dataframe I looped through every player from ‘stats’ for 20 years to get player data for every season. The outer loop filters one player at a time, and after the inner loop goes through one season at a time. All of the games

a player recorded statistics in for that season are then aggregated and compressed into a singular row and the features are a season mean, aside from biometric data. Additionally the inner loop reviews the ‘injury’ dataset for the player and season to determine if the player was injured or not. If the player was injured during that season the ‘injury severity’ column value is set to 1, but if the player was uninjured during the season the value is set to zero. After the inner loop finished executing the aggregated season statistics for the player was appended to a final dataframe that the model would be trained with. As a result every player had 20 rows of data even if they did not play 20 seasons. These fake seasons contained NaN values so after the loops finished I dropped the rows with NaN values, thus eliminating seasons without real data. The final data frame composed 8 features that were used to train the model and the class distribution for the target feature was almost evenly split.

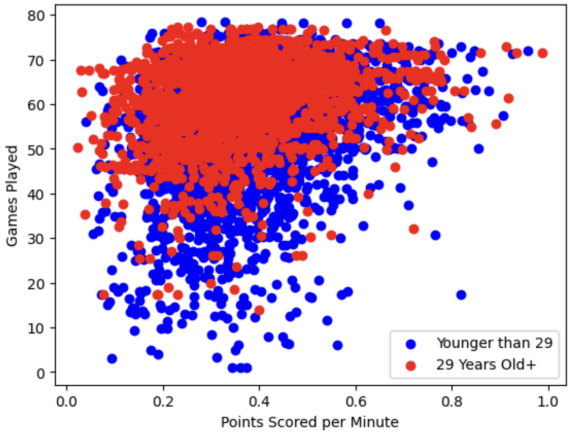

Figure 2: Points per Minute vs Games Played

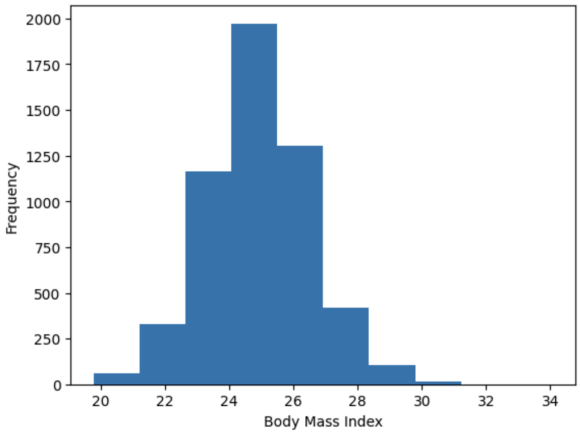

Figure 3: Histogram of BMI

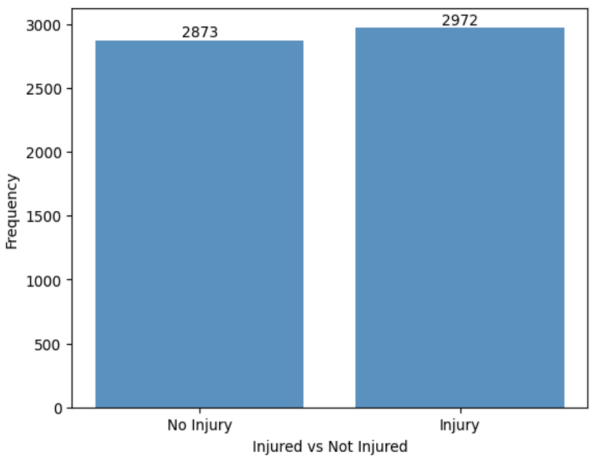

Figure 4: Histogram of Seaons with Injuries and Seaons without Injuries

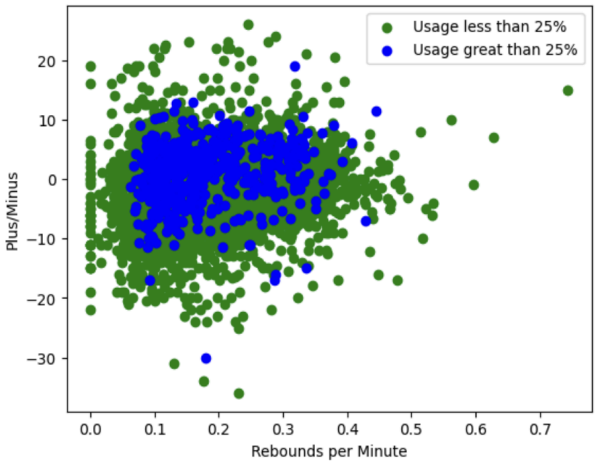

Figure 5: Rebounds per Minute vs Plus/Minus

Input features:

- Number of games played
- Age of player
- Body mass index of player
- Average minutes played per game
- The average score differential while a player was in the game
- 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

All of these features were calculated and assigned to the specific player for each specific game they played between 2000 and 2020.

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. A 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. Its use of multiple estimators (decision trees) with bootstrap sampling helps reduce overfitting. The model uses randomized data for each tree to account for outliers and then uses the combinating of all trees to determine which splits are the most optimal. 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 solve 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.

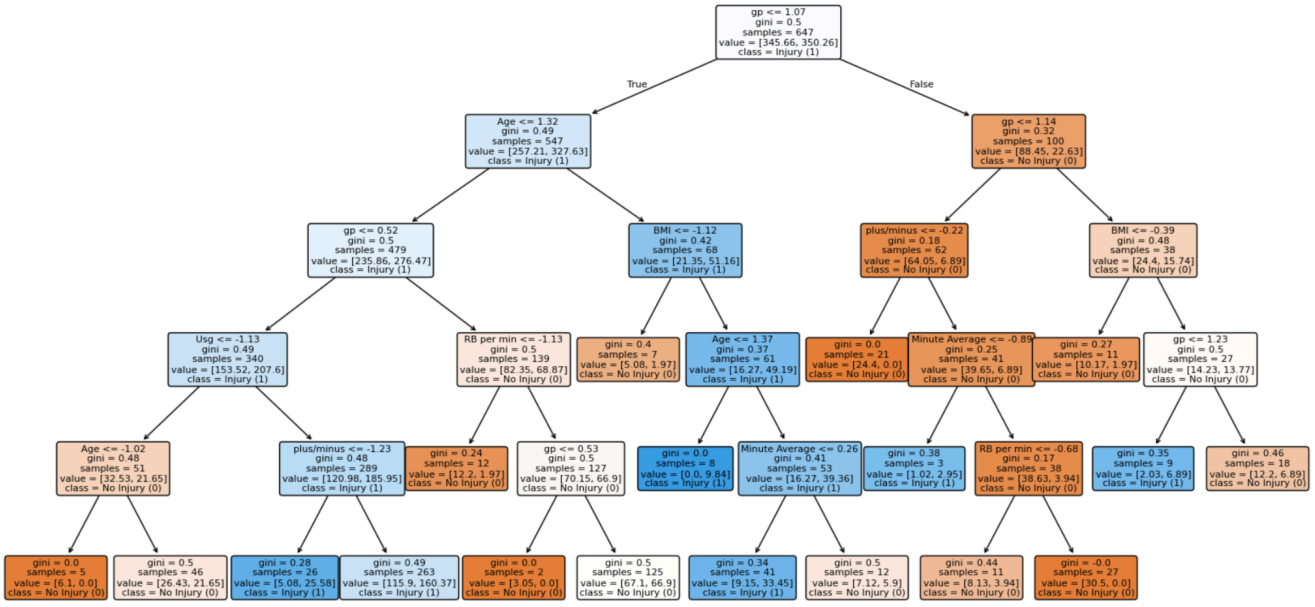

Figure 6: Example Decision Tree from Ensemble

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: [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

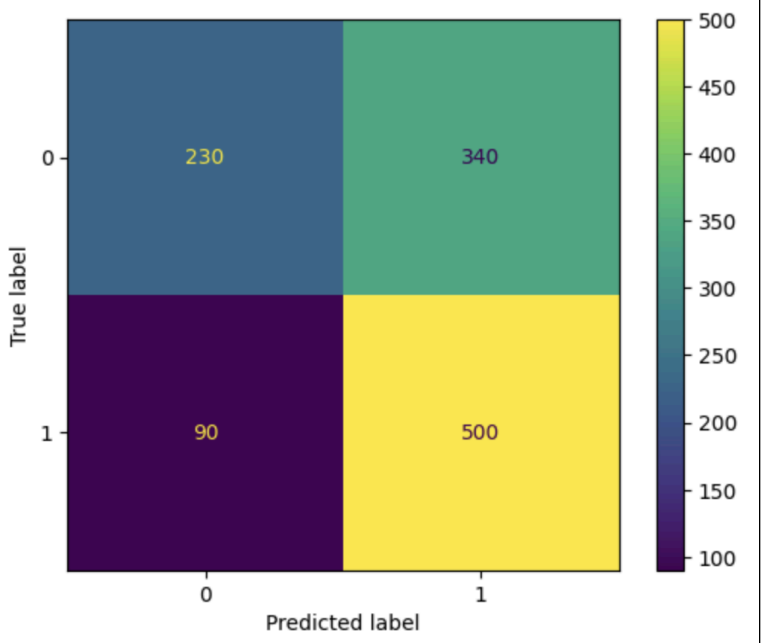

Figure 7: Results from the confusion matrix, displaying the predictions from model testing

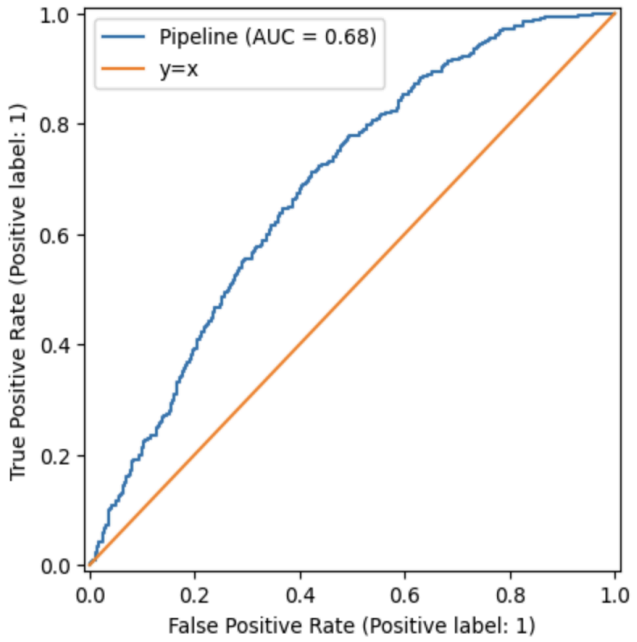

Figure 8: ROC curve for the model, displaying the model's relative success at separating the injured and non-injured player classes

The confusion matrix above 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 in 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

Partial Dependence Plots (PDPs) for each input feature: The 8 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.

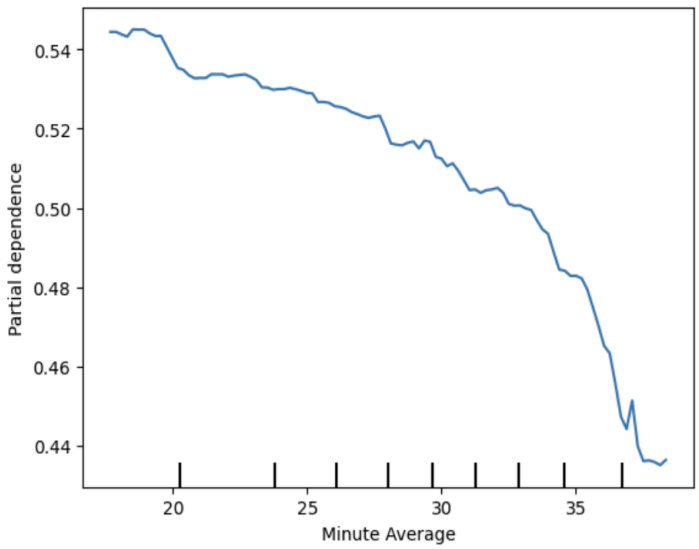

Figure 9: Partial Dependence Plot for Average Minutes Played

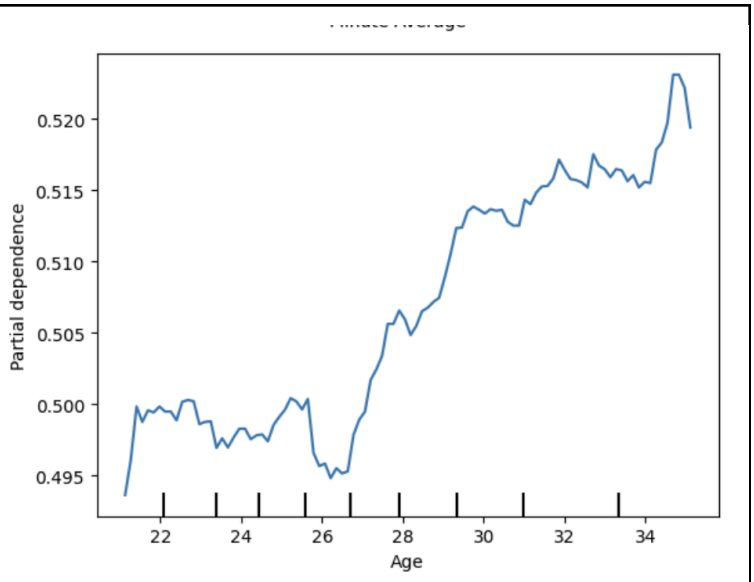

Figure 10: Partial Dependence Plot for Average Age

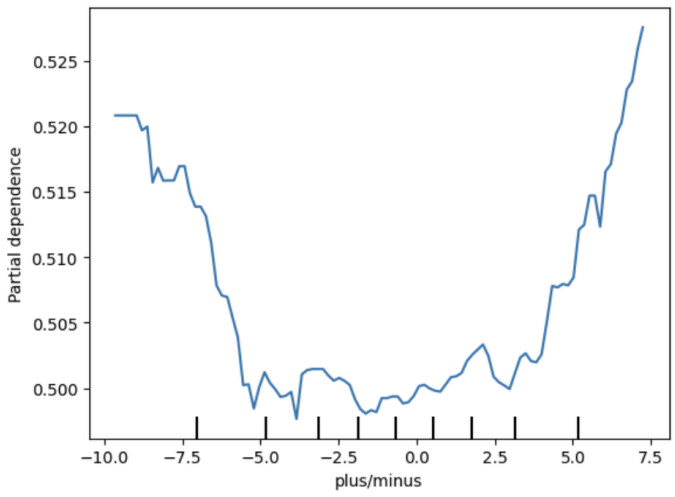

Figure11: Partial Dependence Plot for Plus/Minus

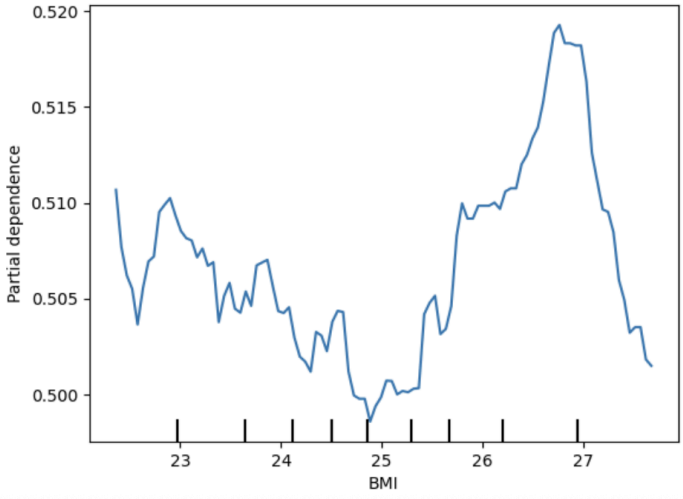

Figure 12: Partial Dependence Plot for Body Mass Index

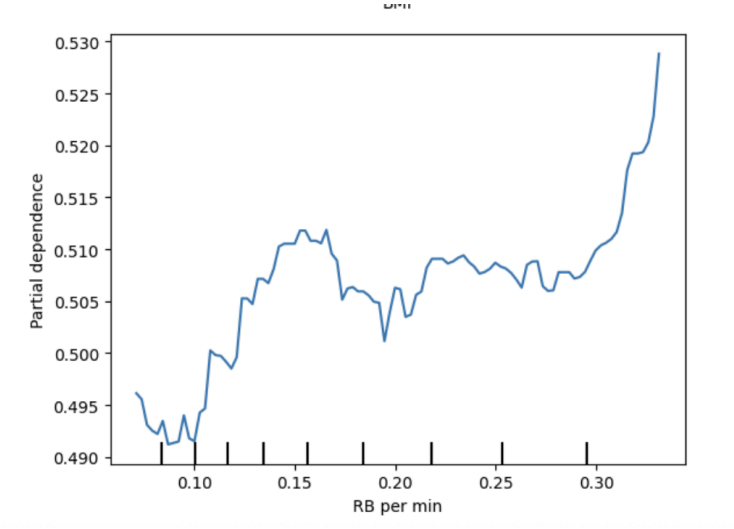

Figure 13: Partial Dependence Plot for Average Rebounds Recorded per Minute

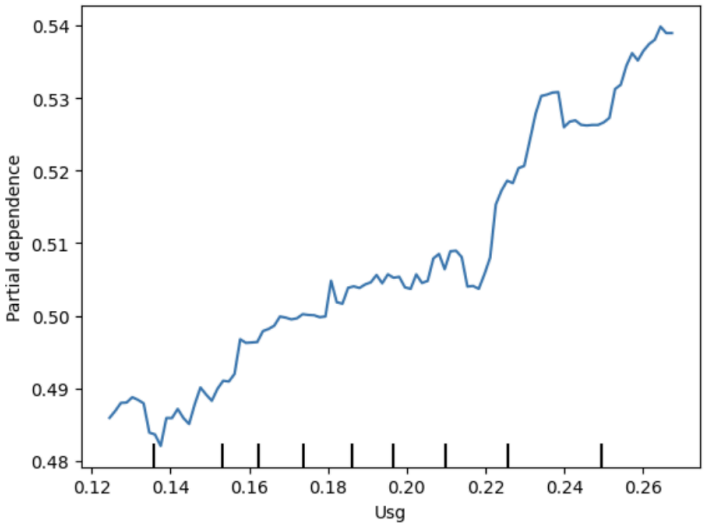

Figure 14: Partial Dependence Plot for Average Usage

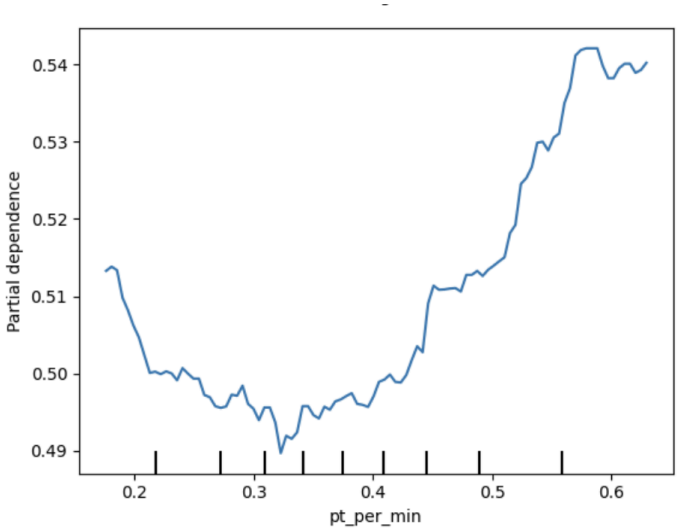

Figure 15: Partial Dependence Plot for Average Points Scored per Minute

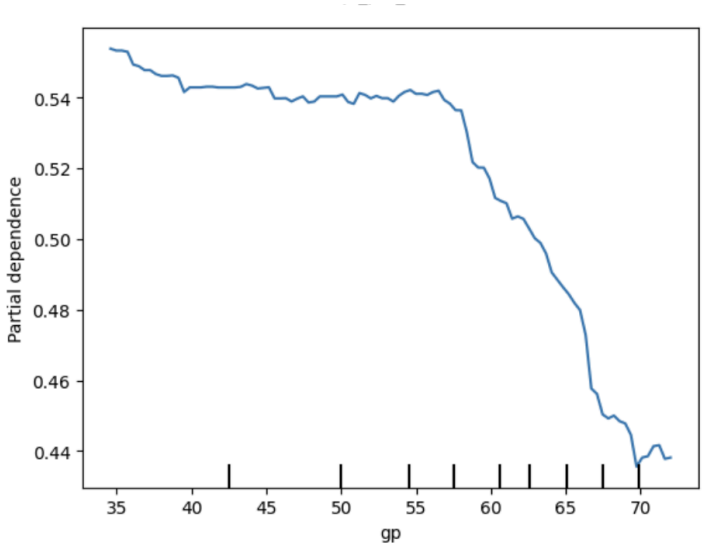

Figure 16: Partial Dependence Plot for Average Games Played

Discussion

From the mean cross validation accuracy and relative success of the confusion matrix it is clear that the model has shortcomings. In order to evaluate if the Random Forest Classifier was the cause of the moderate performance I tried using an Extra Trees Classifier, Gradient Boost Classifier, and Ada Boost Classifier. For each classifier I used, I modified the training parameters—max depth, class weights, max samples—to attempt to improve the model, but had no success. None of the alternative estimators yielded improved model results. The mean cross

validated accuracy from the other models hovered in the range of 55-62% and the Ada Boost model almost always predicted injury (1). In another attempt to improve my results I fit the data using a Support Vector Machine to differentiate the classes. Once again the results were moderate, which led me to suspect the data was ill suited for this problem.

The data I gathered, plus the features I created and used to train the model are not enough to distinguish between players who get injured and those who do not. The input features used for the model cover a wide range of categories including play style, team contribution, game impact, body health, and level of play. The features were chosen based on their high potential to contribute to injury, but there was too much overlap between the injured and non-injured players for every feature. The strong overlap between the classes is apparent in the PDPs.

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. Overall, none of the features exhibit a very strong effect on the model's predictions, as the predicted probabilities generally remain centered around 0.5. This finding reinforces my claim that the data is the cause of the model's moderate performance and is ill suited to predict injury.

The model determined that the number of games played was the most important feature, yet this is not reflected in the PDP. The plot shows a trend that more games played corresponds to a prediction of no injury; however the y-axis shows that the impact of this feature is hardly distinguishable, with its lowest average value at 0.44. This value of 0.44 while closer to 0 than 1 is much closer to 0.5 which corresponds to a feature indistinguishable by class on average. It is unlikely that the feature importance results of this model are robust and will need to be further verified by future work.

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 not the cause of the moderate performance because the use of other estimators and a myriad of alternative parameters did not improve performance. It is clear from the partial dependence plots that none of the input features were strongly distinguishable based on class, and are 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.

More influential feature data will be needed for a future project that attempts to predict injury outcome accurately. The modern era of the NBA tracks numerous biometric player statistics that are still novel. The data available during the 2000 NBA season versus the 2025 NBA season is astronomically different, and that data might be needed to address this problem. With player tracking technology data is available that records player player dunks, including the distance of the jump, the time spent in air, the power of liftoff, and the verticality. A future project may be able to succeed by capitalizing on modern data features that are strongly differentiable between players. If the necessary computing power was available, modern player tracking could be applied to recordings of games from 2000-2020 and the modern data could be added to the dataframe of this project. This additional data and further feature manipulation would potentially enable this model to better predict injury outcome.

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-seasons