Predicting NBA Player Injuries: A Data Driven Approach

Süleyman Yolcu [*1] **Emirhan Utku** [*1]

Abstract

Injuries are a pervasive concern in professional sports, with the National Basketball Association (NBA) offering a unique context for examining this issue due to its high intensity and physical demands. This project aims to develop a predictive model for NBA player injuries using machine learning techniques. By integrating historical injury data with player performance metrics, we strive to identify patterns and factors that contribute to injury risk. Our preliminary findings suggest that decision tree models show promise in accurately classifying injury-prone players, thereby paving the way for improved injury prevention strategies, enhanced player longevity, and more informed team decision-making.

1. Introduction

Injuries in the National Basketball Association (NBA) significantly impact players, teams, and the league as a whole. From a player's perspective, injuries can derail promising careers and shorten playing longevity. For teams, unanticipated injuries to key athletes can drastically alter season trajectories and championship aspirations. At the league level, high-profile injuries can affect viewership, revenue, and the overall quality of competition.

Over the past decade, the growing sophistication of sports analytics has ushered in new approaches to understanding and mitigating injury risk. Machine learning (ML) and data-driven techniques now enable the integration of large, heterogeneous datasets—including game statistics, biometric measurements, and training loads—to identify hidden correlations and patterns associated with injury occurrences. This data-centric methodology has far-reaching implications. By revealing potential risk factors, ML models can help coaches and medical staff optimize training regimens, manage player workloads, and implement targeted interventions tailored to each athlete's profile.

In the NBA's fast-paced environment, factors such as minutes played, back-to-back games, and accumulated fatigue can expose players to increased injury risk. Consequently, developing robust predictive models that account for such situational stressors represents a pivotal step toward reducing downtime and extending playing careers. Ultimately, the capability to predict which players are more susceptible to injuries and when they might occur has the potential to revolutionize player management strategies, minimize disruptions to team rosters, and sustain higher levels of competition for fans.

This project aims to develop a machine learning–based framework to predict NBA player injuries using historical injury records combined with in-game performance metrics. By identifying patterns and correlates of injury risk—such as workload, minutes played, player physical attributes, or specific types of injuries—coaches, trainers, and team decision-makers could proactively manage player rest, recovery protocols, and conditioning programs. Ultimately, the goal is to enhance player well-being, guide strategic rotational decisions, and improve fan engagement by maintaining healthier rosters throughout the season.

2. Related Work

The field of sports injury prediction has witnessed significant advances in recent years, largely due to expanded data availability and sophisticated machine learning algorithms. Within the NBA context, a notable contribution comes from Farghaly and Deshpande (Farghaly & Deshpande, 2024) , who applied models such as Random Forest, Gradient Boosting, and K-Nearest Neighbors to anticipate injuries. Their dataset emphasized physical measurements (e.g., player height, weight, and agility tests) as well as historical injury records. Notably, their Random Forest classifier attained an accuracy of 92.47%, illustrating the strength of ensemble methods in handling complex interactions among anthropometric data and prior injury history.

Although our approach shares similarities with Farghaly and Deshpande's work—particularly in combining historical injury data with player-specific attributes—it departs from theirs by incorporating a broader range of in-game performance variables, such as minutes played, pace of play, and distance covered. By capturing game-by-game trends, our study aims to identify acute stressors (e.g., back-to-back

games or consistently elevated playing time) that may precipitate injuries.

Methodologies from other sports domains further inform our work. Majumdar et al. (Majumdar et al., 2022). explored injury prediction in football by examining both external (velocity, acceleration) and internal (heart rate, fatigue levels) training-load variables. Despite basketball's unique demands—greater game frequency, smaller playing area, and high-intensity bursts—this research underscores important challenges inherent to sports injury modeling, such as data imbalance (injuries vs. healthy samples) and limited multi-season data. The authors also highlight the practical value of interpretable models like Decision Trees, which can reveal clear decision paths (e.g., "Minutes Played ¿ 35 → High Risk") and thereby facilitate load management or targeted preventive measures.

Taken together, these studies illustrate the broader promise of ML-based solutions in professional sports. Farghaly and Deshpande's NBA-focused methods emphasize how historical injury records and physical metrics can boost prediction accuracy, while Majumdar et al. address generalizable concerns about data imbalance and model interpretability. Our project builds on these insights by integrating both physical and performance-driven variables to capture a more holistic view of injury risk. By tailoring this approach to the NBA's high-paced environment, we aim to ensure that our predictive models remain accurate, generalizable, and ultimately beneficial to players, teams, and league stakeholders.

3. Methodology

3.1. Data Sources and Overview

This study leverages two complementary datasets to investigate injury occurrence among NBA players and develop predictive models: Historical Injuries (2010–2020) (nba, n.d.a): A comprehensive dataset of 27,105 entries documenting player injuries. Each entry includes the date of injury, team changes (acquired or relinquished), and descriptive notes about the injury. This source captures a broad view of injury patterns and trends over a decade. Player-Season Performance (2013–2023) (nba, n.d.b): A dataset of 5,578 player-season records that integrate game participation (minutes played, pace, usage), physical attributes (height, weight), workload indicators (distance covered, average speed), and injury outcomes (days missed, injury type). Among these records, 1,214 contain detailed injury annotations (e.g., date of injury, return date, injury type). Together, these datasets provide both the historical context of injuries and granular performance metrics, laying a robust foundation for injury risk analysis and prediction.

3.2. Data Preprocessing

We began by removing irrelevant or non-injury records from the historical injuries dataset (2010–2020). For instance, entries in the notes column that merely indicated an activation or return from injury were excluded. We also normalized descriptive notes by removing extraneous text such as placed on IL or relinquished to ensure only valid injury instances remained.

Algorithm 1 Filter Non-Injury Keywords

```
Define non_injury_keywords = ['activated', 'returned',
'lineup', 'cleared', 'ready']
Filter Rows:
injuries_df ← injuries_df[Notes.str.contains
                (non_injury_keywords
                ,case=False, na=False) == False]
```

Dates were converted to appropriate datetime formats, and corresponding NBA seasons were mapped. Text-based columns with categorical variables (e.g., type of injury, team) were standardized to reduce duplication. We introduced a binary label (Injured) to indicate whether a player incurred at least one injury within a given season. Additionally, we flagged indefinite absences based on text patterns such as "indefinitely," helping capture severity in subsequent analyses.

3.3. Dataset Integration

We next merged the filtered historical injuries dataset with the performance dataset (2013–2023) by aligning player identifiers, team, and season keys. This enabled the creation of a unified table that linked each player's performance metrics with any corresponding injury history. Duplicate rows (where the same player-season existed in both datasets) were removed or consolidated to maintain consistency.

Algorithm 2 Merge Datasets on Player, Team, and Season

```
Merge Datasets:
merged_df ← pd.merge(
    stats_df, injuries_df,
    how='left',
    left_on=['PLAYER_NAME', 'SEASON', 'TEAM'],
    right_on=['Relinquished', 'Season', 'Team']
)
```

3.4. Feature Engineering

Several derived features were generated to support downstream modeling and analysis. Numeric or categorical indicators such as 'body part', 'injury type', and 'days missed' were extracted from textual notes and standardized. Existing columns (e.g., minutes played, and usage rate) were

transformed or scaled as needed to facilitate model training. Height and weight were retained to examine their relationship with injury propensity, particularly for lower-body versus upper-body injuries. For players with available tracking data (e.g., distance covered, average speed), additional workload-based features were incorporated to explore the link between physical exertion and injury risk.

3.5. Exploratory Analysis

Before predictive modeling, we conducted descriptive statistics and generated visualizations to highlight the most common injury types (Figure 1), body region vulnerability (Figure 3), and player characteristics and injuries. Knee and ankle injuries were particularly common, as seen in Figure 2. This is consistent with the high-impact nature of basketball movements. Lower-body injuries (knees, ankles, feet) were associated with repetitive jumping and cutting. Explorations and Figure 5 showed that taller, heavier players often sustained knee and back injuries, while shorter guards were prone to foot or ankle issues , reflecting agility demands.

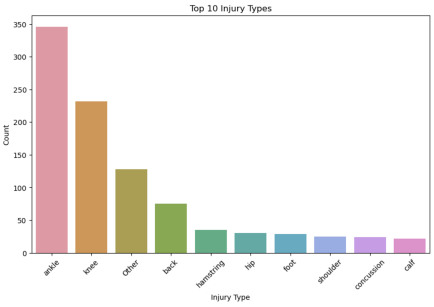

Figure 1. Top 10 Injury Types observed in the dataset.

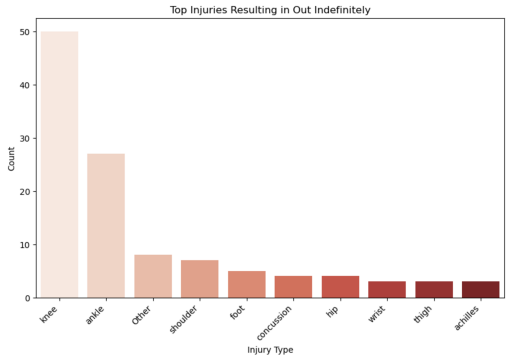

Figure 2. Top injuries resulting in indefinite absences.

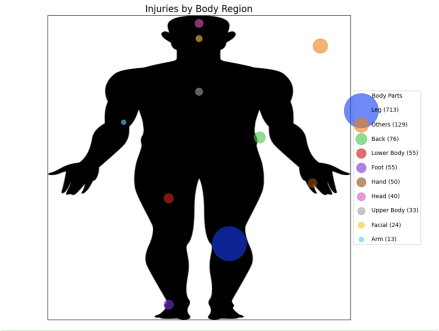

Figure 3. Injuries by body region

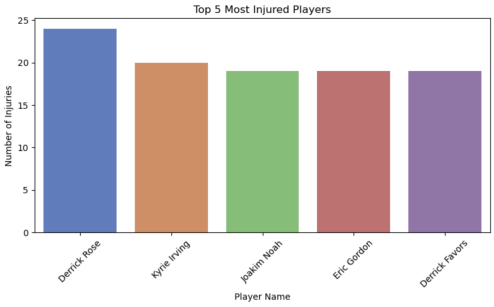

Figure 4. Top 5 Most Injured Players

As can be seen from Figure 5, Favors and Noah show a heavy concentration of knee and back issues, likely tied to their larger builds and front-court roles. Rose, Irving, and Gordon also have recurrent knee injuries but exhibit more varied distributions (e.g., ankle, foot, elbow), reflecting the dynamic nature of guard play and frequent change-of-direction movements.

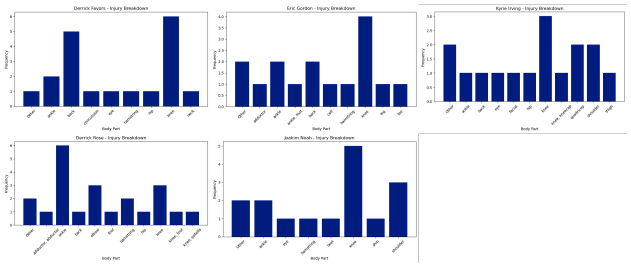

Figure 5. Injury Breakdown by Body Part.

The Figure 6 underscores that taller, heavier players (e.g., Favors) experience significant stress on lower-body joints, while shorter guards (e.g., Rose) still sustain numerous injuries due to high-intensity gameplay. This reinforces the idea that both physical attributes and on-court roles drive susceptibility to injury.

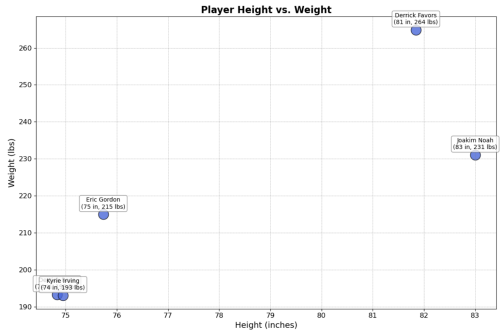

Figure 6. Player Height vs. Weight

3.6. Modeling Approach

We built a uniform preprocessing pipeline for numeric and categorical variables, ensuring consistent handling of missing values, scaling, and one-hot encoding.

Algorithm 3 Random Forest Pipeline

```
# Numerical and categorical transformations
numerical_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])
categorical_transformer = Pipeline([
    ('imputer',
     SimpleImputer(strategy='most_frequent')),
    ('onehot',
     OneHotEncoder(handle_unknown='ignore'))
])
preprocessor = ColumnTransformer([
    ('num', numerical_transformer, numerical_cols),
    ('cat', categorical_transformer, categorical_cols)
])
# Random Forest pipeline
random_forest_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=100, random_state=42))
])
```

Several classification algorithms were then evaluated: Random Forest, emphasized robust performance with parameters such as "n_estimators" and "max_depth". Decision Tree, offered transparency in predictions but was prone to overfitting. K-Nearest Neighbors (KNN), tested various "n_neighbors", "weights", and "metric" settings, though balancing classes was crucial. Support Vector Machine (SVM), explored linear versus RBF kernels, tuning C and gamma to manage margin and misclassification. Logistic Regression, provided a baseline, focusing on interpretability and class-imbalance considerations.

3.7. Model Training and Evaluation

We split the combined dataset into training (80%) and test (20%) sets, while preserving class distribution via stratified sampling. Each model underwent the same training and evaluation routine: Fit Model (Train on the 80% subset), Predict (Generate predictions on the test data), Assess Performance (Measure accuracy, precision, recall, and F1-score. Inspect confusion matrices and classification reports to identify misclassification patterns).

Algorithm 4 Model Training and Evaluation

```
# Fit model
random_forest_pipeline.fit(X_train, y_train)
# Evaluate model
def evaluate_model(name, model, X_test, y_test):
    y_pred = model.predict(X_test)
    print(f"— {name} —")
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Confusion Matrix:", confusion_matrix(y_test, y_pred))
    print("Classification Report:",
    classification_report(y_test, y_pred))
evaluate_model("Random Forest",
random_forest_pipeline, X_test, y_test)
```

3.8. Hyperparameter Tuning and Validation

To enhance model stability and handle potential class imbalance, we implemented 5-fold cross-validation for each algorithm. For Random Forest, we performed grid search over parameters like "n_estimators", "max_depth", and "min_samples split"; for Decision Trees, we used randomized search to optimize depth and splitting criteria. Optimal hyperparameters were selected based on highest cross-validated accuracy.

Algorithm 5 GridSearch for Random Forest

```
param_grid =
    'classifier__n_estimators': [100, 200, 300],
    'classifier__max_depth': [10, 20, 30],
    'classifier__min_samples_split': [2, 5, 10]

grid_search = GridSearchCV(
    random_forest_pipeline,
    param_grid,
    cv=5,
    n_jobs=-1,
    scoring='accuracy'
)
grid_search.fit(X_train, y_train)
best_rf_model = grid_search.best_estimator_
```

4. Results

All experiments were carried out by splitting the data into a training set of 3,418 examples and a test set of 855 examples, each containing 24 features. The split followed an 80:20 ratio with stratified sampling to maintain a balanced representation of injuries. Five different classification algorithms—Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM)—were then evaluated on the test data to establish baseline performance before hyperparameter tuning.

The initial evaluation revealed that K-Nearest Neighbors reached an accuracy of 0.7754, as seen in the confusion matrix (Figure 15) showing relatively even detection of the majority (non-injured) class but moderate misclassification of injured players. Its weighted F1-score of 0.62 indicated a fair overall balance, although the recall of 0.66 for the injured class highlighted potential for improvement (Figure 7).

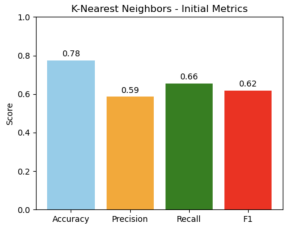

Figure 7. K-Nearest Neighbors Initial Metrics.

Logistic Regression attained an accuracy of 0.7637 (Figure 8), with high precision for the non-injured class but substantially lower recall for the injured class (0.32), suggesting a propensity to favor the majority class under default settings. The Support Vector Machine achieved an accuracy of 0.8047 (Figure 9) and a notably higher precision (0.75) than recall (0.45) for the injured class, implying that it more confidently flagged true positives at the expense of missing some injuries.

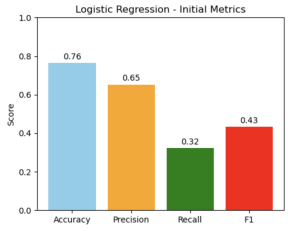

Figure 8. Logistic Regression Initial Metrics.

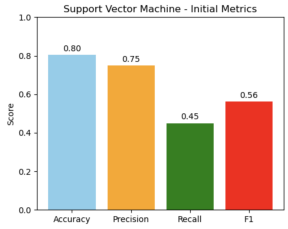

Figure 9. Support Vector Machine Initial Metrics.

Decision Tree classification stood out with an accuracy of 0.8667 (Figure 10) and an impressively high recall of 0.91 for the injured class, indicating a strong ability to detect positive cases. However, its decision boundaries risked overfitting, as Decision Trees often memorize training patterns if left unconstrained.

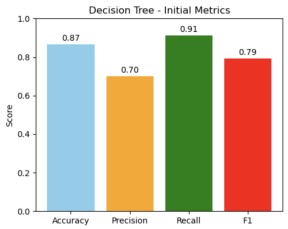

Figure 10. Decision Tree Initial Metrics.

Random Forest delivered the best performance among all initial models, reaching an accuracy of 0.9497 (Figure 11), with balanced gains in precision (0.91) and recall (0.90) for the injured class. By aggregating multiple decision trees, Random Forest demonstrated better generalization capability than a single tree classifier.

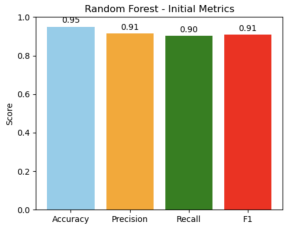

Figure 11. Random Forest Initial Metrics.

For a side-by-side comparison of Accuracy, Precision, Recall, and F1-score across these initial models, refer to Figure 12.

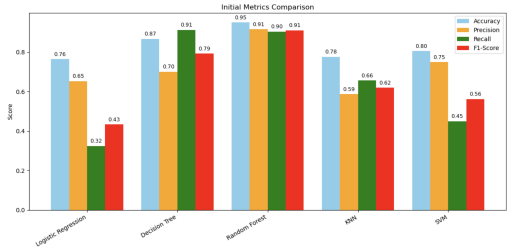

Figure 12. Accuracy Comparison: Initial, CV, and Best (HP).

To further assess each model's robustness and reduce overfitting, a 5-fold cross-validation procedure was performed. Logistic Regression showed a mean CV accuracy of 0.7393 ± 0.0105, which led to a best-tuned cross-validation accuracy of 0.7440 after optimizing parameters such as regularization (C) and penalty type. The Decision Tree saw a mean CV accuracy of 0.8435 ± 0.0109, with its best configuration involving a deeper tree depth (up to 40) and minimal constraints on splits or leaf nodes, preserving a strong 0.8435 CV accuracy after tuning. Random Forest again surpassed the other methods, sustaining an impressive mean CV accuracy of 0.9213 ± 0.0064 and improving to 0.9266 through the selection of a larger number of estimators and a maximum depth of 30 while disabling bootstrap sampling. K-Nearest Neighbors scored a mean CV accuracy of 0.7753 ± 0.0100, but extensive hyperparameter tuning—particularly around the distance metric and the number of neighbors—boosted its best achievable CV accuracy to 0.8698. The Support Vector Machine's mean CV accuracy of 0.7809 ± 0.0187 rose to 0.8534 with an RBF kernel and appropriately tuned gamma and C values.

As illustrated in Figure 13, a comparison of initial accuracy, cross-validation accuracy, and best hyperparameter performance highlights these improvements across all models. Furthermore, Figure 14 provides a detailed breakdown of accuracy, precision, recall, and F1-score during the cross-validation process, showcasing each model's strengths and weaknesses.

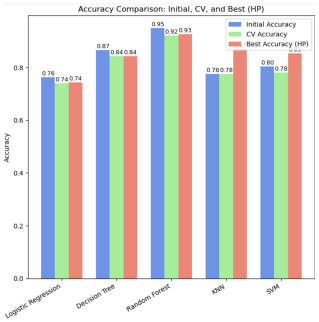

Figure 13. Accuracy Comparison across Initial, Cross-Validation, and Best Hyperparameter Configurations.

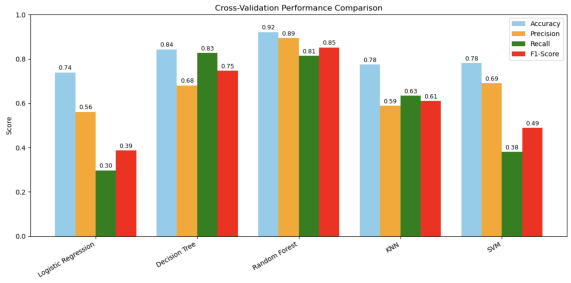

Figure 14. Cross-Validation Performance Comparison of Different Models.

Examining these cross-validation results revealed important trade-offs. For instance, Decision Trees and Random Forests exhibited high recall for the minority (injured) class, though Decision Trees risked memorizing training examples without strong regularization. Models such as Logistic Regression and SVM provided more interpretable or theoretically grounded decision boundaries but tended to struggle in identifying all injured cases when using default parameters. Class imbalance evidently influenced these results; models often favored the majority (non-injured) class, prompting the need for additional balancing strategies or cost-sensitive training to improve recall and precision for injuries.

Despite Random Forest's strong performance, several limitations must be noted. The injury labels within the data can contain ambiguities or partial information, since not all injuries are consistently documented and timelines of recovery may vary. The data also come from multiple seasons, raising concerns about temporal shifts in player performance, training regimens, and league styles. Advanced workload metrics such as distance covered and average speed were not uniformly available, which introduces potential biases or missing patterns in certain subsets of the data. Furthermore, although Random Forest and other ensemble methods achieve superior results, they can be opaque in practice, requiring additional interpretability tools to reveal which features and interactions most strongly contribute to predicted risks.

4.1. Hyperparameter Tuning Results

The hyperparameter tuning process revealed distinct outcomes for each classification algorithm—Logistic Regression, Decision Tree, Random Forest, SVM, and K-Nearest Neighbors (KNN).

Logistic Regression, best tuned with $C = 0.1$, L2 regularization, and the "liblinear" solver, achieved a mid-seventies cross-validation accuracy, indicating that purely linear boundaries cannot fully capture the complexity of this injury data. A single Decision Tree set to a maximum depth of 40, with minimal leaf and split constraints, reached the

mid-eighties but ran a higher risk of overfitting.

Random Forest emerged as the top performer, pushing cross-validation accuracy into the low nineties. By using 300 trees with a maximum depth of 30, no bootstrapping, and minimal splitting constraints, it capitalized on the ensemble approach to better generalize. SVM, employing an RBF kernel with $\gamma = 0.1$ and $C = 10$, also delivered strong performance in the mid-eighties, while KNN—tuned to 27 neighbors with Manhattan distance and distance-based weighting—claimed a close second place.

Table 1. Best Hyperparameters and Cross-Validation Accuracies for Each Model

Model	Best Parameters	Best CV Accuracy
Logistic Regression	C: 0.1, Penalty: l2, Solver: liblinear	0.7440
Decision Tree	Criterion: gini, Max Depth: 40, Min Samples Leaf: 1, Min Samples Split: 2	0.8435
Random Forest	Bootstrap: False, Max Depth: 30, Min Samples Leaf: 1, Min Samples Split: 2, n_Estimators: 300	0.9266
K-Nearest Neighbors	Metric: manhattan, n_Neighbors: 27, Weights: distance	0.8698
Support Vector Machine	Kernel: rbf, Gamma: 0.1, C: 10	0.8534

These results highlight Random Forest's dominance and the surprising competitiveness of KNN. They also show that tree ensembles and instance-based methods, when carefully tuned, outperform single-tree or linear models in capturing the intricate nature of sports injury data.

5. Conclusion

This project aimed to predict NBA player injuries by integrating two complementary datasets and employing a range of classification models. The historical injury data spanning 2010 to 2020 offered a broad perspective on the frequency and nature of injuries, while the player-season performance records from 2013 to 2023 provided granular information on game participation, physical measurements, and workload variables. Through systematic data cleaning, feature engineering, and merging processes, a unified view of each player's risk factors was constructed, capturing both their background injury history and current performance metrics.

The experimental results demonstrated that ensemble methods, particularly Random Forest, delivered the highest predictive accuracy. In the initial tests, Random Forest achieved an accuracy of 94.97% on the held-out test set, with similarly strong precision and recall for the injured class. Cross-validation and hyperparameter tuning further refined the model's performance, ensuring that the gains were not solely due to overfitting. Although single Decision Trees also showed promise, their propensity to overfit to training data was more evident without proper constraints. Logistic Regression, SVM, and K-Nearest Neighbors performed reasonably well but often struggled with class imbalance, especially in detecting true positives for injury cases. The final outcomes suggest that combining ensemble learning with a robust feature set can enhance predictive power in sports analytics.

Overall, this work highlights the value of combining diverse data sources and ensemble-based machine learning to predict injuries in professional basketball. By continuing to refine the data pipeline, expanding available metrics, and employing more sophisticated analytical techniques, sports organizations and medical teams can improve early detection and prevention strategies for athlete injuries.

5.1. Suggestions For Further Results

Despite encouraging results, limitations remain. Data coverage varied among players, and injury labels were occasionally inconsistent or incomplete, suggesting potential underreporting or ambiguity. Future directions could include incorporating additional contextual variables, such as game scheduling intensity or advanced biometric tracking, to uncover further injury risk factors. Employing deeper interpretability methods (e.g., SHAP values) could provide insights into how the model prioritizes features like minutes played or historical injury patterns. Exploring advanced machine learning frameworks, such as gradient boosting or deep learning, might better capture complex player dynamics across seasons, though richer datasets with finer temporal granularity would be required.

References

Nba injuries 2010–2018 dataset, n.d.a. Dataset available at https://www.kaggle.com/datasets/ghopkins/nba-injuries-2010-2018.

Nba player stats and injured data from 2013 to 2023, n.d.b. Dataset available at https://www.kaggle.com/datasets/icliu30/nba-player-stats-and-injured-data-from-1[illegible]3.

Farghaly, O. and Deshpande, P. Leveraging machine learning to predict national basketball association player injuries. In *Proceedings of the 2024 IEEE International Workshop on Sport, Technology and Research (STAR)*, Lecco, Italy, 2024. doi: 10.1109/STAR62027.2024.10636005.

Majumdar, A., Bakirov, R., Hodges, D., et al. Machine learning for understanding and predicting injuries in football. *Sports Medicine – Open*, 8:73, 2022. doi: 10.1186/s40798-022-00465-4.

A. Confusion Matrices for Initial Model Performance

This appendix contains the confusion matrices for the initial performance of the evaluated classification models, highlighting the distribution of true positive, true negative, false positive, and false negative predictions.

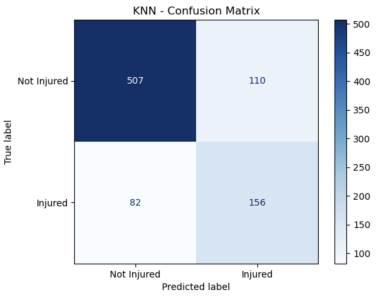

Figure 15. K-Nearest Neighbors Confusion Matrix.

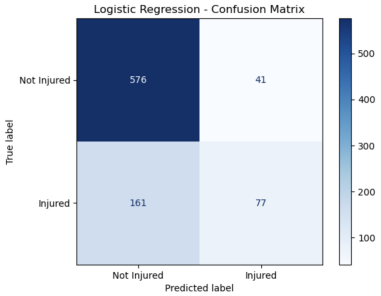

Figure 16. Confusion Matrix for Logistic Regression (Initial Performance).

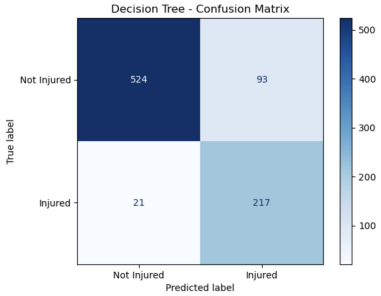

Figure 17. Confusion Matrix for Decision Tree (Initial Performance).

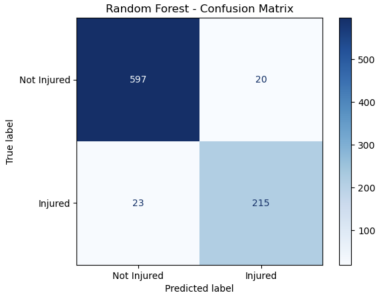

Figure 18. Confusion Matrix for Random Forest (Initial Performance).

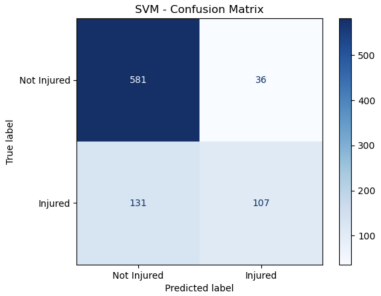

Figure 19. Confusion Matrix for Support Vector Machine (Initial Performance).