**Injury Prediction Model for Professional Sports Team**

**Introduction**

As part of my portfolio, I developed an injury prediction model using historical data of players' biometric and workload metrics. This model aims to predict the likelihood of injuries, providing insights to the medical team for proactive injury prevention and player health management. The following report details the methodology, implementation, and outcomes of this project.

**Data Overview**

The dataset comprises various attributes, including player age, weight, height, previous injury history, training intensity, and recovery time. The target variable, Likelihood\_of\_Injury, indicates the probability of a player sustaining an injury.

Key Columns:

* **Player\_Age**: Age of the player.
* **Player\_Weight**: Weight of the player.
* **Player\_Height**: Height of the player.
* **Previous\_Injuries**: Binary indicator of whether a player had previous injuries.
* **Training\_Intensity**: Measure of training load or intensity.
* **Recovery\_Time**: Average recovery time after training sessions or injuries.
* **Likelihood\_of\_Injury**: Target variable indicating the likelihood of injury.

**Methodology**

**Data Preprocessing**

1. **Data Cleaning**: Missing values were handled using forward filling, ensuring the dataset's completeness.
2. **Feature Engineering**: Interaction features such as Age\_Weight, Age\_Height, and Intensity\_Recovery were created to capture more complex relationships between player attributes and injury risk.
3. **Scaling**: Numerical features were normalized using StandardScaler to ensure uniformity in scale, facilitating better model performance.

**Model Development**

A Gradient Boosting Classifier was selected due to its ability to handle complex datasets and provide high predictive accuracy. To optimize the model's performance, hyperparameters were fine-tuned using Grid Search with a cross-validation approach.

* **Hyperparameters Tuned**:
  + n\_estimators: Number of boosting stages.
  + learning\_rate: Learning rate shrinks the contribution of each tree.
  + max\_depth: Maximum depth of the individual trees.

**Results**

**Model Performance**

* **Best Parameters**: n\_estimators=200, learning\_rate=0.1, max\_depth=4
* **Classification Report**:
  + Precision, recall, and F1-score metrics were used to evaluate model performance.
* **ROC-AUC Score**: The model achieved an ROC-AUC score of [Insert Score], indicating good discrimination between injured and non-injured players.

**Feature Importance**

The model identified key features influencing injury risk, which include:

* **Training\_Intensity**: Significantly impacts the likelihood of injury.
* **Previous\_Injuries**: A crucial predictor of future injury risk.
* **Age\_Weight** and **Age\_Height**: Interaction features that help understand the combined effect of age, weight, and height on injury risk.

**Insights and Recommendations**

The injury prediction model provides valuable insights into the factors contributing to player injuries. Key recommendations for the medical and training staff include:

1. **Monitoring Training Intensity**: Adjust training loads to prevent overtraining, especially for players with a history of injuries.
2. **Individualized Recovery Programs**: Tailor recovery programs based on player-specific data to optimize recovery times and reduce injury risk.
3. **Focus on At-Risk Players**: Pay special attention to players identified as having a higher likelihood of injury, implementing targeted prevention strategies.

**Conclusion**

This project demonstrates the potential of data analytics in enhancing player health management and injury prevention strategies in professional sports. The developed model provides actionable insights that can significantly benefit the team's medical and training staff, contributing to better player performance and longevity.

**Future Work**

Future improvements could include integrating more comprehensive data such as in-game performance metrics, psychological factors, and environmental conditions. Additionally, exploring advanced machine learning techniques like deep learning could further enhance predictive accuracy.

Code

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.preprocessing import StandardScaler, PolynomialFeatures

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import classification\_report, roc\_auc\_score

# Load the dataset

file\_path = ‘injury\_data.csv'

df = pd.read\_csv(file\_path)

# Feature Engineering: Example of creating interaction features

df['Age\_Weight'] = df['Player\_Age'] \* df['Player\_Weight']

df['Age\_Height'] = df['Player\_Age'] \* df['Player\_Height']

df['Intensity\_Recovery'] = df['Training\_Intensity'] / df['Recovery\_Time']

# Define the target (y) and features (X)

y = df['Likelihood\_of\_Injury']

X = df.drop('Likelihood\_of\_Injury', axis=1)

# Normalize numerical features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Split the data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# Use Gradient Boosting Classifier with hyperparameter tuning

param\_grid = {

'n\_estimators': [100, 200, 300],

'learning\_rate': [0.01, 0.1, 0.2],

'max\_depth': [3, 4, 5]

}

gbc = GradientBoostingClassifier(random\_state=42)

grid\_search = GridSearchCV(estimator=gbc, param\_grid=param\_grid, scoring='roc\_auc', cv=5, n\_jobs=-1)

grid\_search.fit(X\_train, y\_train)

# Best model from grid search

best\_model = grid\_search.best\_estimator\_

# Model Evaluation

y\_pred = best\_model.predict(X\_test)

y\_pred\_proba = best\_model.predict\_proba(X\_test)[:, 1]

print("Best Parameters:", grid\_search.best\_params\_)

print(classification\_report(y\_test, y\_pred))

print('ROC-AUC Score:', roc\_auc\_score(y\_test, y\_pred\_proba))

# Provide Insights

feature\_importances = pd.DataFrame(best\_model.feature\_importances\_, index=X.columns, columns=['importance']).sort\_values('importance', ascending=False)

print(feature\_importances)