Predicting Soccer Injuries with Machine Learning

Module 4: Prototype Solution

Jennifer Lawless
DASC 9311: Data Science Project
October 22, 2025

Project Timeline

Week 1 (9/1-9/7): Dataset acquisition, initial exploration, environment setup

Week 2 (9/8-9/14): Review of existing Kaggle solutions, methodology research

Weeks 3-4 (9/15-9/28): EDA, feature analysis & engineering, data preprocessing

Today (10/22)

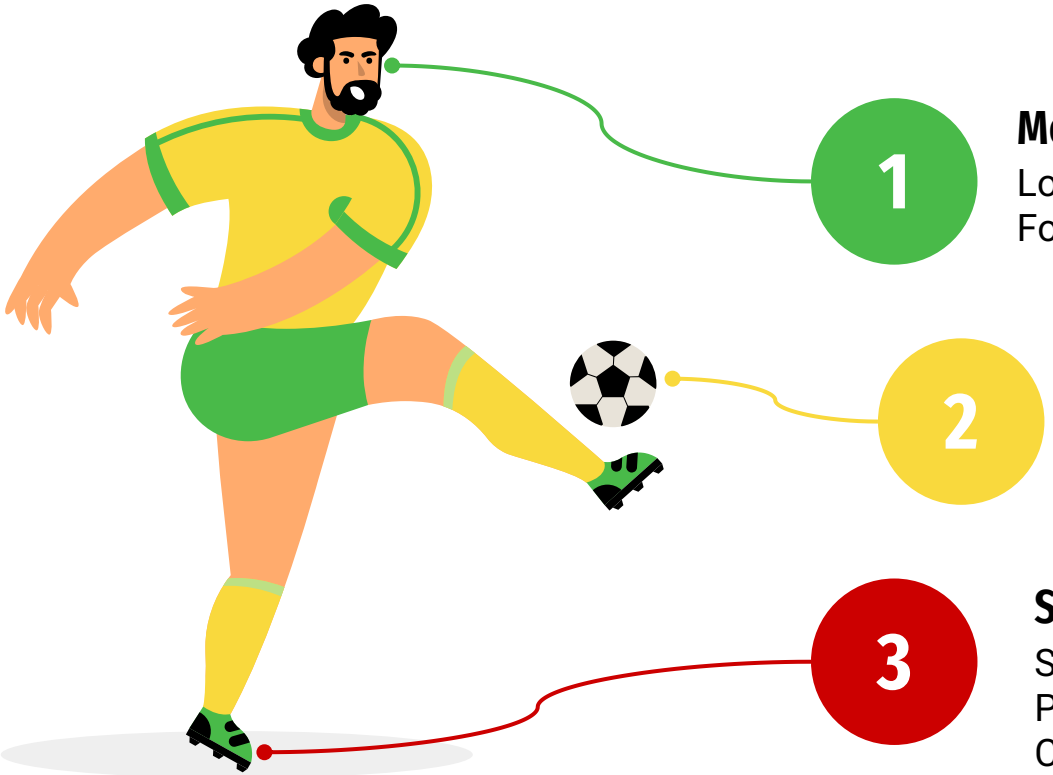
Weeks 5-7.5 (9/29-10/22): Baseline model implementation, initial performance benchmarking

Weeks 7.5-10.5 (10/23-11/5): Advanced model development, model fine-tuning

Weeks 10.5-13 (11/6-11/23): Ensemble model development, model fine-tuning

Weeks 14-15 (11/24-12/7): Final model selection, results analysis, documentation

Baseline Modeling Approach



1

Model Types

Logistic Regression, Random Forest, XGBoost

2

Validation Strategy

Nested cross-validation with outer loop (5-fold) and inner loop (3-fold)

3

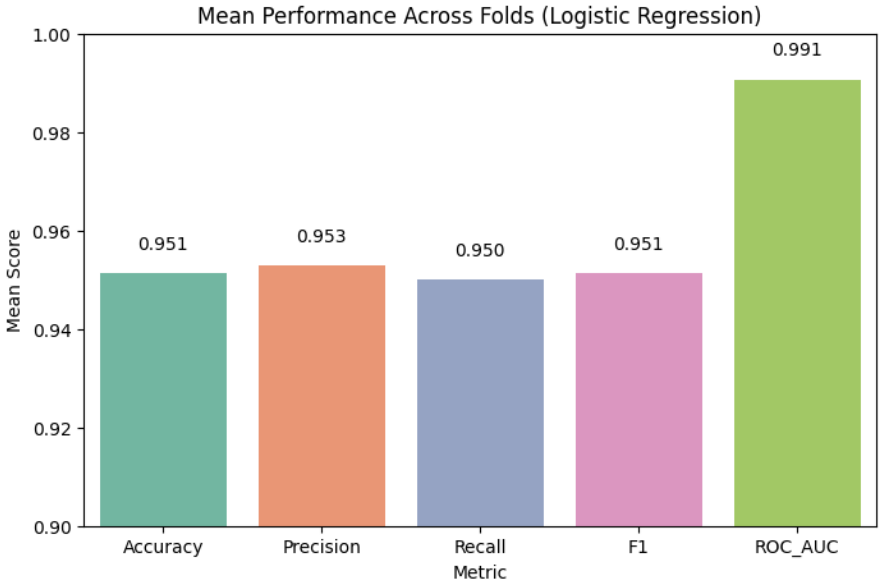
Scoring and Evaluation

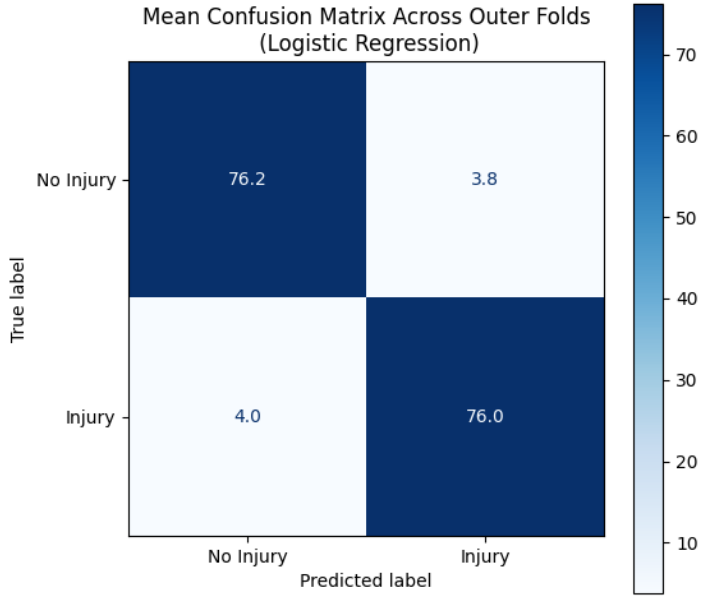
Scoring Metrics: Accuracy, Precision, Recall, F1, ROC-AUC
Confusion matrix, feature importances

Model 1: Logistic Regression

Model Setup

- **Algorithm:** Logistic Regression (binary classification)
- **Solver:** supports L1 & L2 penalties
- **Max Iterations:** 1000 (ensures convergence)
- **Hyperparameters tuned:**
  Regularization strength = [0.01, 0.1, 1, 10]
  Penalty = L1 (Lasso) or L2 (Ridge)

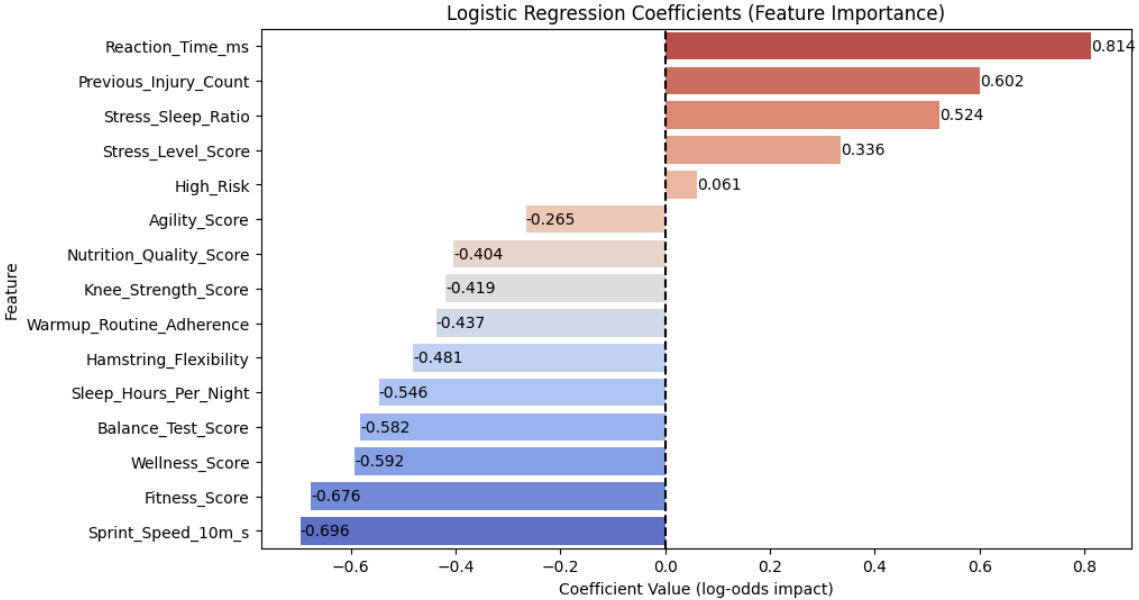


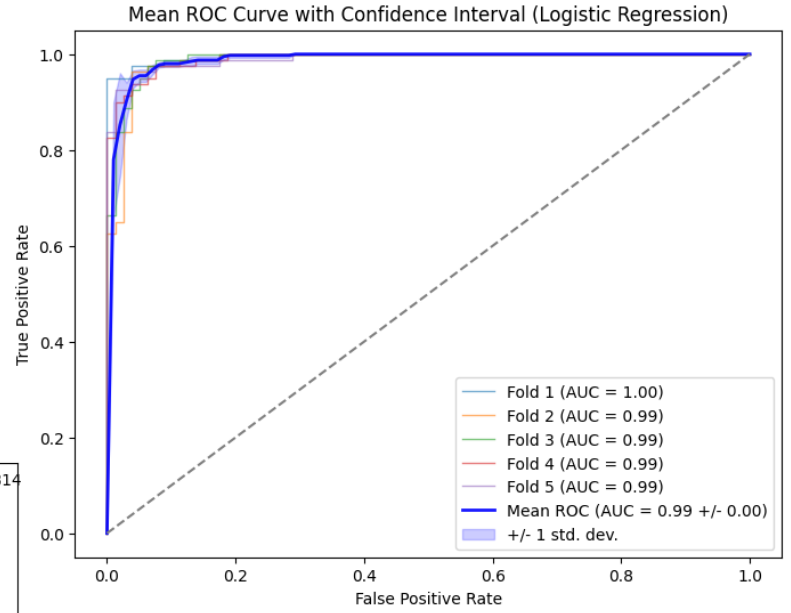


Comparison to Existing Kaggle Solutions (Accuracy Rate)

- My Solution: 0.951
- Kaggle Solution 1: 0.945
- Kaggle Solution 2: 0.950

Model 1: Logistic Regression (cont.)

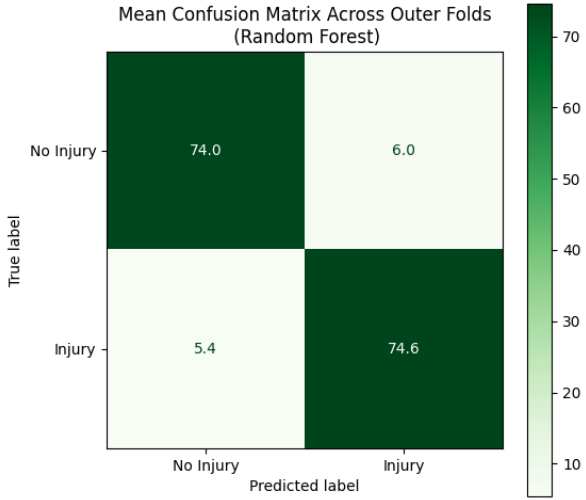


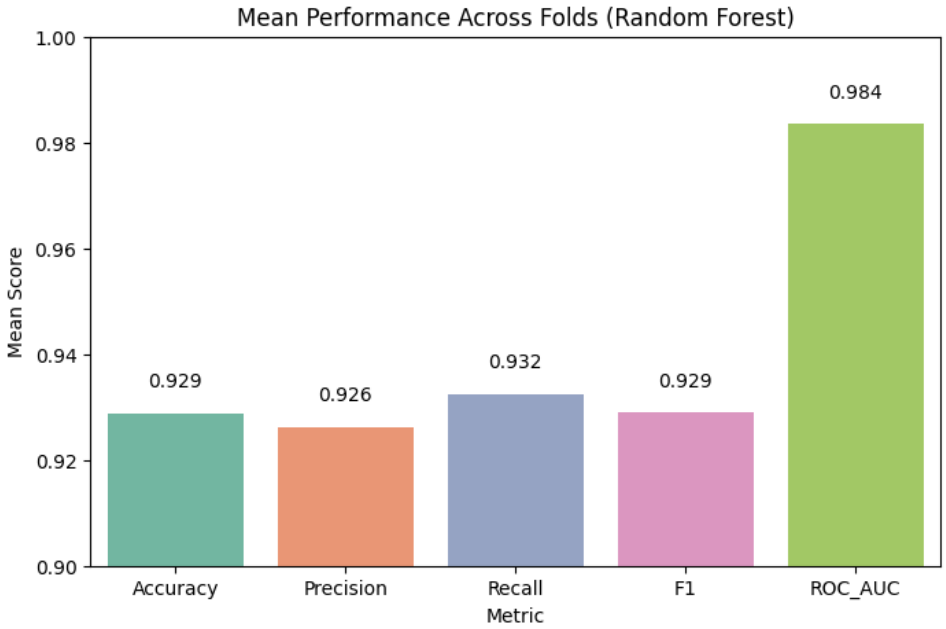


Model 2: Random Forest

Model Setup

- **Algorithm:** Random Forest Classifier (ensemble of decision trees, binary classification)
- **Hyperparameters Tuned:**
  - **Number of trees:** [50, 100, 200]
  - **Maximum depth:** [None, 10, 20, 30]
  - **Minimum samples per split:** [2, 5, 10]
  - **Minimum samples per leaf:** [1, 2, 4]
  - **Max features:** [‘sqrt’, ‘log2’]

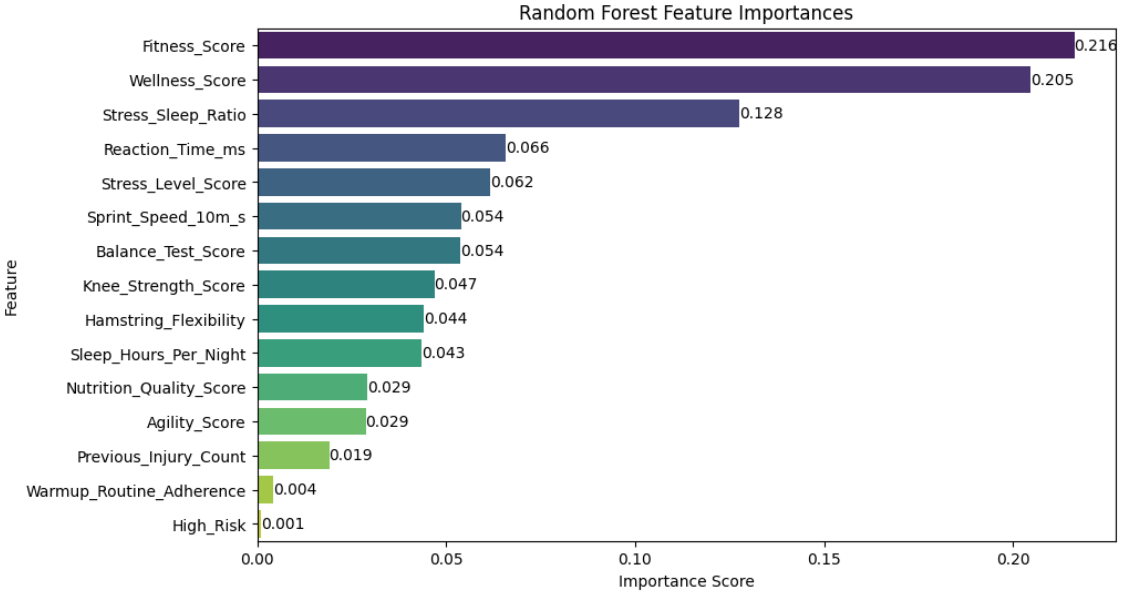


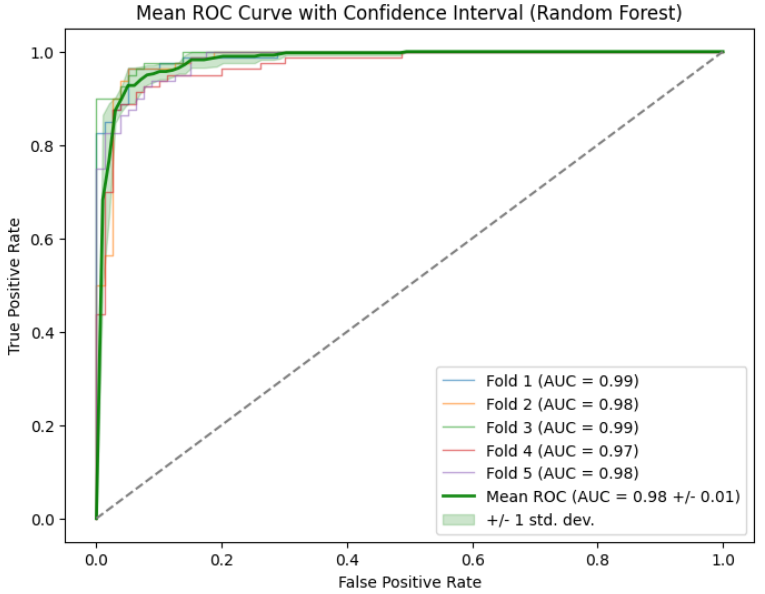


Comparison to Existing Kaggle Solutions (Accuracy Rate)

- My Solution: 0.929
- Kaggle Solution 1: 0.906
- Kaggle Solution 2: 0.963

Model 2: Random Forest (cont.)



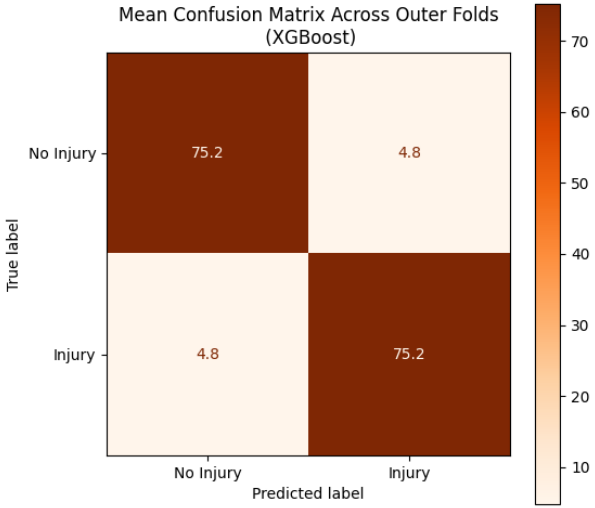


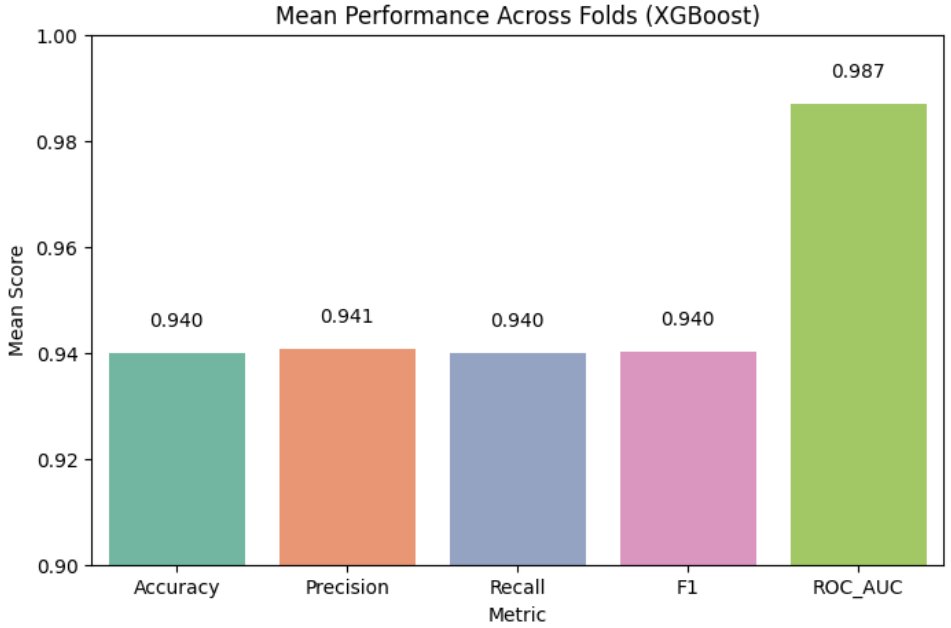
Model 3: XGBoost

Model Setup

Algorithm: XGBoost Classifier (gradient boosting, binary classification)
Hyperparameters Tuned:

- **Learning rate:** [0.01, 0.1, 0.3]
- **Maximum depth:** [3, 5, 7, 10]
- **Number of estimators:** [100, 200, 300]
- **Subsample ratio:** [0.8, 1.0]
- **Column sampling by tree:** [0.8, 1.0]

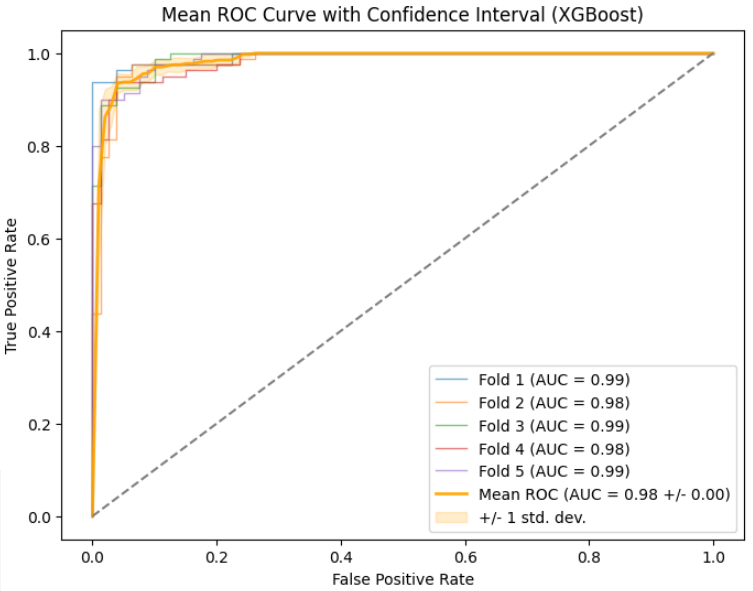


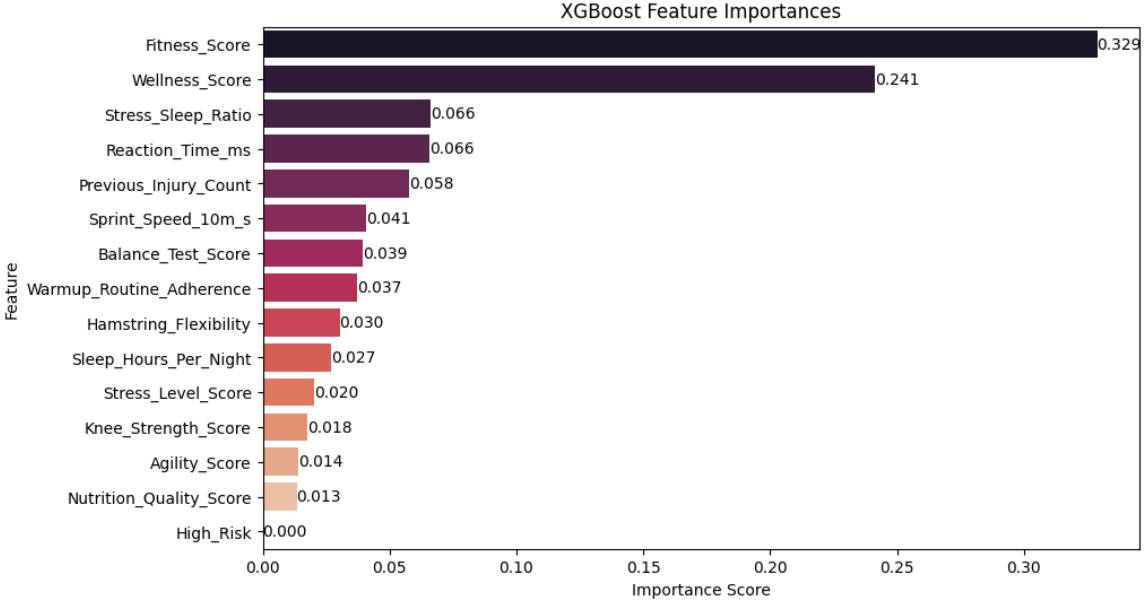


Comparison to Existing Kaggle Solutions (Accuracy Rate)

- My Solution: 0.940
- Kaggle Solution 1: 0.894
- Kaggle Solution 2: 0.969

Model 3: XGBoost (cont.)





Model Comparison

Model	Logistic Regression	Random Forest	XGBoost	Summary
Metrics	**Accuracy:** 0.951 **Precision:** 0.953 **Recall:** 0.950 **F1:** 0.951 **ROC-AUC:** 0.991	**Accuracy:** 0.929 **Precision:** 0.926 **Recall:** 0.932 **F1:** 0.929 **ROC-AUC:** 0.984	**Accuracy:** 0.940 **Precision:** 0.941 **Recall:** 0.940 **F1:** 0.940 **ROC-AUC:** 0.987	LR best overall, then XGB, then RF
Key Feature	Reaction_time_ms (+0.814)	Fitness_Score (+0.216)	Fitness_Score (+0.329)	Engineered features more dominant in trees
Strengths	Simple, interpretable, Highest ROC-AUC	Handles non-linearity and feature interactions	Fast, regularized, good generalization	All strong, but LR is slightly better
Limitations	Assumes linearity, less robust to interactions	Slower training, lowest accuracy	Potential overfitting if not tuned	Tree models underperformed vs. Kaggle solutions

Shortcomings and Goals for Improvements

Shortcomings		Goals for Improvements	
✖	Impact of feature engineering (caused underperformance vs. Kaggle solutions)	✔	Expand hyperparameter tuning with RandomSearchCV for broader ranges
✖	Computational challenges	✔	Feature selection to reduce noise from engineered features
✖	Dataset limitations	✔	Optimize runtime
✖	Model-specific issues		

Next Steps

Improve baseline models

- Improve Random Forest and XGBoost models to beat existing Kaggle solutions

Advanced and ensemble model development

- Build TabNet, GPC, SVM, LightGBM, Stacking Ensemble, Voting Classifier, Bayesian Model Averaging models

Model Fine-Tuning & Evaluation

- Fine-tune the models
- Evaluate and analyze each model
- Determine the best-performing model

Documentation

- Document all steps taken
- Write the final report and prepare for presentations

Thank You!