🔟 Week 10: Machine Learning for Injury Risk

This code sets up, trains, and evaluates two machine learning models (Logistic Regression and Random Forest) to predict injury risk.

```
# Week 10: Machine Learning for Injury Risk Prediction


import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, confusion_matrix


# --- Data Loading (Assumes 'bulls_injury_data.csv' was generated in the setup cell) ---

df = pd.read_csv('bulls_injury_data.csv')


# --- 1. Define Features (X) and Target (y) ---

X = df[['Minutes_Load', 'HighSpeedRuns', 'JumpLoad', 'HeartRate', 'SleepHours', 'PreviousInjury']]

y = df['InjuryNextWeek']


# --- 2. Split the Data (80% Train, 20% Test) ---

X_train, X_test, y_train, y_test = train_test_split(

  X, y, test_size=0.2, random_state=42
```

```python
)

# --- 3. Scale Features (Essential for Logistic Regression) ---
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)



# --- 4. Train and Evaluate Logistic Regression ---
print("\n--- Logistic Regression ---")
log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train_scaled, y_train)
y_pred_log = log_reg.predict(X_test_scaled)
print("📊 Classification Report:")
print(classification_report(y_test, y_pred_log, zero_division=0)) # zero_division=0 handles the imbalance

# --- 5. Train and Evaluate Random Forest ---
print("\n--- Random Forest ---")
# Using original X data (no scaling needed for tree-based models)
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print("📊 Classification Report:")
print(classification_report(y_test, y_pred_rf, zero_division=0))
```

```python
# --- 6. Visualize Feature Importance ---

feat_importance = pd.Series(rf.feature_importances_, index=X.columns)

plt.figure(figsize=(8,5))

feat_importance.sort_values().plot(kind='barh', color='red')

plt.title('Feature Importance - Injury Risk Prediction (Bulls)')

plt.show()


# --- 7. Confusion Matrix (Final Visualization) ---

cm = confusion_matrix(y_test, y_pred_rf, labels=[0, 1]) # Ensure correct labels are included

sns.heatmap(cm, annot=True, fmt='d', cmap='Reds', xticklabels=['No Injury', 'Injury'],
yticklabels=['No Injury', 'Injury'])

plt.title('Confusion Matrix - Random Forest')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()
```