```
# 1. Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
import streamlit as st
import os
# 2. Load Data
DATA_PATH = "collegiate_athlete_injury_dataset.csv"
if not os.path.exists(DATA_PATH):
    st.error(f" Dataset not found: {DATA_PATH}")
else:
    df = pd.read_csv(DATA_PATH)
    df.columns = [col.strip() for col in df.columns]  # Remove accidental spaces
     #3. Data Cleaning
    st.write("  Data Cleaning Summary")
    st.write(df.isnull().sum())  # Show missing values
    #Convert to categorical
    for col in ['Gender', 'Position']:
        df[col] = df[col].astype('category')
    # Encode categorical variables
    le_gender = LabelEncoder()
    le_position = LabelEncoder()
    df['Gender_num'] = le_gender.fit_transform(df['Gender'])
    df['Position_num'] = le_position.fit_transform(df['Position'])
  #  4. Exploratory Data Analysis (EDA)
    st.write("Injury Case Distribution")
    fig1, ax1 = plt.subplots(figsize=(6,4))
    sns.countplot(x='Injury_Indicator', data=df, ax=ax1)
```

```
ax1.set_title('Injury Case Distribution')
st.pyplot(fig1)
st.write("Fatigue by Injury Status")
fig2, ax2 = plt.subplots(figsize=(6,4))
sns.boxplot(x='Injury_Indicator', y='Fatigue_Score', data=df, ax=ax2)
ax2.set_title('Fatigue by Injury Status')
st.pyplot(fig2)
# 5. Feature Engineering
features = [
    'Age','Height_cm','Weight_kg','Training_Intensity','Training_Hours_Per_Week',
    'Recovery_Days_Per_Week','Match_Count_Per_Week','Rest_Between_Events_Days',
    'Fatigue_Score','Performance_Score','Team_Contribution_Score',
    'Load_Balance_Score','ACL_Risk_Score','Gender_num','Position_num'
]
X = df[features]
y = df['Injury_Indicator']
# Normalize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# 6. Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
)
# 7. Model Building
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
# 8. Model Evaluation
st.write("Model Evaluation Results")
report = classification_report(y_test, y_pred, output_dict=True)
report_df = pd.DataFrame(report).transpose()
st.dataframe(report_df)
cm = confusion_matrix(y_test, y_pred)
```

```python
st.write("**Confusion Matrix:**")
st.write(cm)
roc_score = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])
st.write(f"**ROC AUC Score:** {roc_score:.3f}")
# 9. Streamlit App - Prediction Section
st.write("Athlete Injury Risk Prediction")
uploaded_file = st.file_uploader(" Upload Athlete Data CSV", type="csv")
if uploaded_file is not None:
    input_df = pd.read_csv(uploaded_file)
    for col in ['Gender', 'Position']:
        input_df[col] = input_df[col].astype('category')
        # Apply same encoders
    input_df['Gender_num'] = le_gender.transform(input_df['Gender'])
    input_df['Position_num'] = le_position.transform(input_df['Position'])
    X_input = input_df[features]
    X_input_scaled = scaler.transform(X_input)
    risk_pred = model.predict(X_input_scaled)
    risk_prob = model.predict_proba(X_input_scaled)[:, 1]
    input_df['Injury_Risk_Prediction'] = risk_pred
    input_df['Injury_Risk_Probability'] = risk_prob
    st.write("Prediction Results")
    st.dataframe(input_df)
    st.success(" Predictions generated successfully!")
st.info("This tool uses a Random Forest model trained on athlete workload and health metrics.")
```