**GROUP B MEMBERS**

BCS-05-0159/2022 AKUMU ALLAN OTIENO

BCS-05-0106/2022 HELLEN MWANGI

BCS-05-0102/2022 NYAWIRA GITARI LYNNE

BCS-05-0159/2022 kathuli Yusuf katuvya

BCS-05-0126/2022 DUNCAN MUSYOKI

Bcs-05-0179/2022 Ochieng Paul

BCS-05-0173/2022 Moses Kuria

BCS-03-0012/2023 Somoire Tom

Bcs-05-0174/2022 Mark Njenga

BCS-05-0112/2022 Stacey atulo

BCS-05-0121/2022 Nyamari Victor

Beverly Mudogo Beverly Mudogo

**Project Overview: Soccer Match Outcome Prediction Using Machine Learning**

**Goal:** We are building a machine learning model to predict the outcomes of soccer matches. The model predicts whether a home team will win, lose, or draw based on various match statistics.

**1. Data Preprocessing and Feature Engineering**

We began by preparing the soccer match dataset for use in machine learning. This involved the following steps:

* **Collecting match statistics**: We collected data such as goals scored, shots, shots on target, fouls, yellow cards, red cards, possession, corners, and offsides for both the home and away teams.
* **Creating features**: For the features, we focused on various match statistics, including goals, shots, possession, and other team actions like fouls and cards. These are the factors that help the model predict match outcomes.
* **Data Scaling**: To ensure that the features were on the same scale (important for most machine learning models), we applied feature scaling using StandardScaler. This normalizes the features so that each has a mean of 0 and a standard deviation of 1.

**2. Building the Machine Learning Model**

We selected the **Random Forest Classifier** as our machine learning model. The steps included:

* **Model Selection**: A Random Forest model was chosen because it can handle both numeric and categorical data, and it is robust to overfitting.
* **Training the Model**: The model was trained on a dataset where the target variable was the match outcome (Home Win, Home Loss, or Draw). We split the data into training and testing sets to evaluate the model’s performance.
* **Hyperparameter Tuning**: Random Forest has several hyperparameters, such as the number of trees (n\_estimators), that can be tuned. We used grid search and cross-validation to optimize these hyperparameters.
* **Model Evaluation**: We evaluated the model using various metrics such as accuracy, precision, recall, and confusion matrix to ensure the model was performing well.

**3. Making Predictions and Output Formatting**

Once the model was trained and tuned, we proceeded with predictions for specific matches. We did this by:

* **Input Data**: We created hypothetical match data (e.g., for matches between Crystal Palace and West Ham, or Manchester United and Liverpool). These data included match statistics like goals, shots, possession, fouls, etc., for both home and away teams.
* **Prediction Logic**: We provided match data as input and fed it into the trained model. The model predicts one of three outcomes:
  + **Home Win (1)**
  + **Away Win (0)**
  + **Draw (2)**

We also ensured that the predicted outcome could be displayed with the team names and the actual outcome to compare whether the prediction was correct.

**4. Model Output Example**

We tested the model by using various match data and predicting outcomes. For example:

* **Crystal Palace vs West Ham**: We input the match statistics into the model, and it predicted the outcome based on the provided data.
* **Manchester United vs Liverpool**: Another match was tested with the home team, Manchester United, predicted to lose to Liverpool based on match statistics (e.g., fewer shots, worse possession).

# Define the missing columns (columns from training data but missing in the input match data)

missing\_columns = ['Offsides Away', 'Offsides Home']

# Add missing columns to the match\_df with default values (e.g., 0)

for col in missing\_columns:

if col not in match\_df.columns:

match\_df[col] = 0 # Or you can fill with a default value, such as the mean value

# Ensure the columns are in the same order as during training

match\_df = match\_df[trained\_columns]

# Scaling the data and making the prediction

match\_df\_scaled = scaler.transform(match\_df)

# Make the prediction

predicted\_outcome = best\_rf\_model.predict(match\_df\_scaled)

# Decode the predicted outcome (0 = Home Loss, 1 = Home Win, 2 = Draw)

predicted\_outcome\_label = le.inverse\_transform(predicted\_outcome)

# Define team names

home\_team = 'Manchester United'

away\_team = 'Liverpool'

# Actual outcome: Home Loss (0)

actual\_outcome = 0 # Home Loss

# Show results

result\_df = pd.DataFrame({

'Home Team': [home\_team],

'Away Team': [away\_team],

'Actual Outcome': [actual\_outcome], # Home Loss encoded as 0 (Away Win for Liverpool)

'Predicted Outcome': [predicted\_outcome\_label[0]]

})

# Display prediction results

print(result\_df)

if actual\_outcome == predicted\_outcome\_label[0]:

print("Prediction was correct!")

else:

print("Prediction was incorrect.")

**5. Troubleshooting Model Predictions**

We encountered some issues where the model predicted **only draws** or **home wins**, regardless of the input data. After reviewing the following factors, we took steps to address them:

* **Class Imbalance**: The model might have been biased due to the training data having more examples of draws and home wins than home losses.
  + **Solution**: We suggested techniques like **resampling** the data (oversampling the minority class or undersampling the majority class), or generating synthetic examples using techniques like **SMOTE**.
* **Model Calibration**: We checked if the model was confident in its predictions by examining the predicted class probabilities using predict\_proba to see if the model was uncertain and always predicting draws.
* **Input Data**: We ensured that the input data was realistic, and the home team was expected to lose in some cases, based on the match statistics.

**6. Final Steps and Deployment**

* We tested several matches, ensuring that the model could predict all three outcomes: **home win**, **away win**, and **draw**.
* The model was evaluated for its accuracy and consistency across different match scenarios.

**Summary**

To summarize:

* We prepared the match data, preprocessed it, and trained a machine learning model (Random Forest) to predict match outcomes.
* We tested the model with different match scenarios, ensuring it could predict home wins, away wins, and draws.
* We identified potential issues with class imbalance and model bias and proposed solutions to address these issues.
* The system is ready for further testing and can be expanded or deployed for real-time predictions in sports applications.