MapReduce-Based Decision Tree for FIFA World Cup Performance Prediction

Aqsat Asif, Harsh Patel

May 2025

1 Introduction, History, Motivation

In the era of big data, sports analytics leverages vast datasets to predict performance and uncover insights. The FIFA World Cup, a global soccer tournament, generates extensive match and player data, making it an ideal case for scalable predictive modeling. Traditional machine learning struggles with large datasets, necessitating distributed frameworks like MapReduce.

1.1 Project Motivation

This project aims to predict team and player performance in the FIFA World Cup using MapReduce for scalable feature extraction and decision trees for interpretable predictions. The motivation is to provide actionable insights for analysts, coaches, and fans while addressing the computational challenges of large-scale sports data.

1.2 Aims and Objectives

The project develops a scalable system for World Cup performance prediction. Objectives include:

- Extract features from FIFA World Cup datasets using MapReduce.
- Train decision trees to classify teams as High/Low Performance and High/Low Goals.
- Evaluate models using accuracy, precision, recall, AUC, and visualizations.
- Identify key performance drivers, such as goals scored or unique scorers.

2 Background

Sports analytics has grown with detailed datasets, enabling predictions of game outcomes and player performance. Decision trees are ideal for their interpretability and effectiveness

with structured data [**?**]. Large datasets like World Cup records require scalable preprocessing, which MapReduce provides through distributed computing [**?**]. Prior work often lacks scalability, a gap this project addresses by integrating MapReduce with decision trees.

2.1 Dataset Description

The project uses two datasets:

- `WorldCupMatches.csv`: Contains ∼4,500 records of matches from 1930–2014, with columns like home/away teams, scores, and match outcomes.
- `WorldCupPlayers.csv`: Includes ∼37,000 records of player events, with fields like player name, team, goals scored, and disciplinary actions.

These datasets provide a comprehensive view of team and player performance, necessitating efficient processing due to their size.

3 Approach and Implementation

3.1 Conceptual Framework

The project employs a pipeline:

- **MapReduce**: Parallel feature extraction for matches played, goals, and outcomes.
- **Decision Trees**: Binary classification using `DecisionTreeClassifier` for team (High-/Low Performance) and player (High/Low Goals) predictions.

3.2 Operational Mechanism

1. **Preprocessing**: Clean datasets using `clean_matches.py` and upload to HDFS via `hdfs_upload.sh`.
2. **MapReduce**: Extract features (e.g., `team_matches.txt`, `player_goals_output.txt`) using scripts like `matches_played_mapper.py`.
3. **Training**: Train decision trees with `decision_teams.py` and `decision_players.py`, outputting `team_predictions.csv` and `player_predictions.csv`.
4. **Evaluation**: Compute metrics and generate visualizations (e.g., ROC curves, confusion matrices).

3.3 Implementation Details

A sample MapReduce mapper counts matches played:

```
#!/usr/bin/env python
import sys
for line in sys.stdin:
    line = line.strip()
    fields = line.split(',')
    home_team = fields[2]
    away_team = fields[3]
    print(f"{home_team}\t1")
    print(f"{away_team}\t1")
```

Listing 1: MapReduce Mapper for Matches Played

The reducer aggregates counts:

```
#!/usr/bin/env python
import sys
current_team = None
current_count = 0
for line in sys.stdin:
    team, count = line.strip().split('\t')
    count = int(count)
    if current_team == team:
        current_count += count
    else:
        if current_team:
            print(f"{current_team}\t{current_count}")
        current_team = team
        current_count = count
if current_team:
    print(f"{current_team}\t{current_count}")
```

Listing 2: MapReduce Reducer for Matches Played

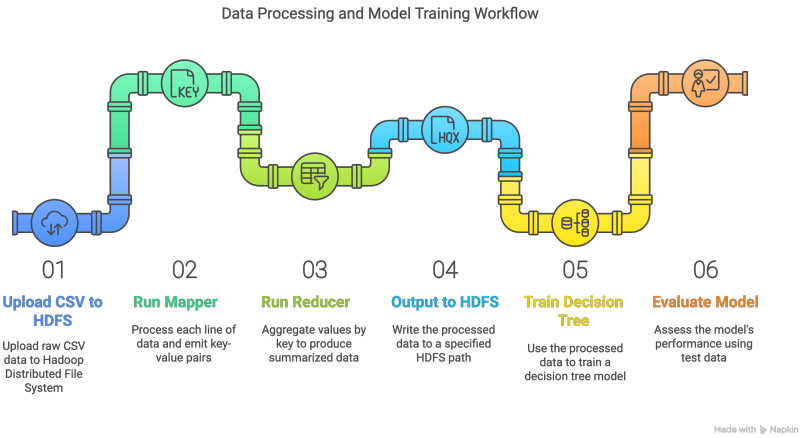


Figure 1: Project Workflow: Data Preprocessing, MapReduce Feature Extraction, Decision Tree Training, and Evaluation

4 Experiment Results and Discussion

4.1 Evaluation Metrics

Model performance is evaluated using:

- **Accuracy**: Fraction of correct predictions.
- **Precision**: Fraction of predicted positives that are correct.
- **Recall**: Fraction of actual positives identified.
- **AUC**: Area under the ROC curve, measuring discriminatory power.

4.2 Team Model

The team decision tree splits on `avg_goals_scored` and `draws`, achieving pure leaves (Gini 0) in training (Figure 2).

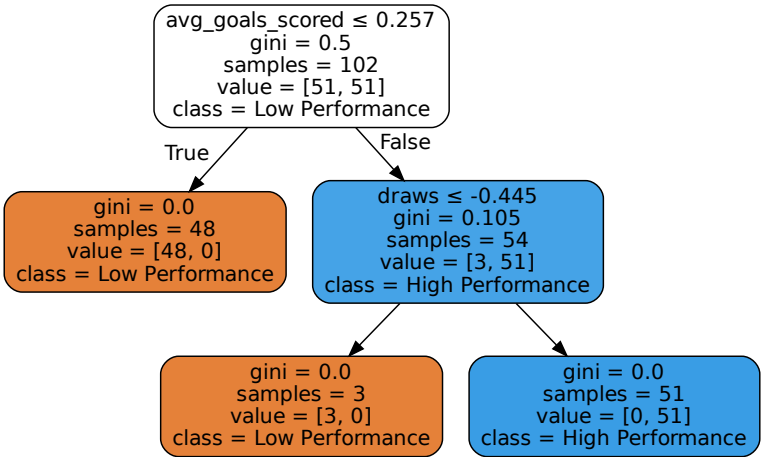
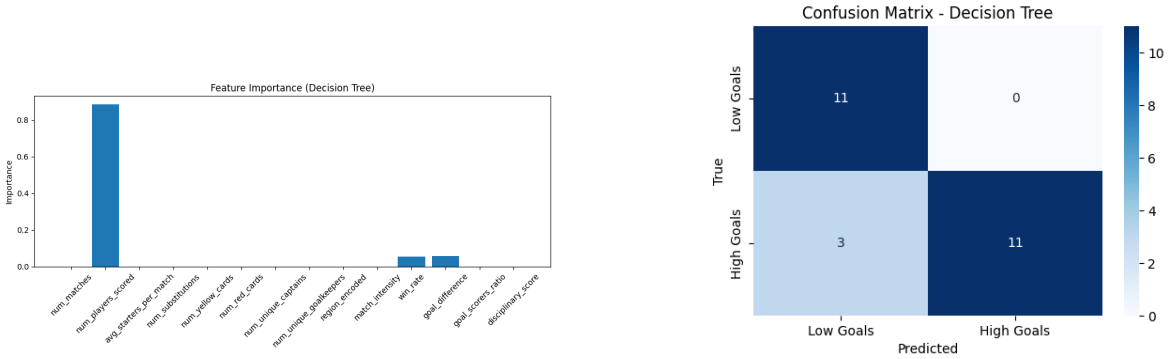


Figure 2: Team Decision Tree Structure

4.3 Player Model

The player decision tree prioritizes `num_players_scored` (importance ∼0.85), with minor roles for `goal_difference` and `disciplinary_score` (Figure 3).



(a) Feature Importance

(b) Confusion Matrix

Figure 3: Player Model Visualizations

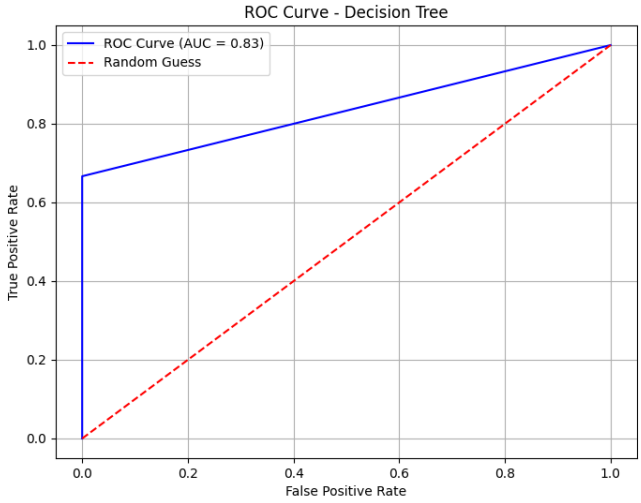


Figure 4: Team Model ROC Curve (AUC = 0.83)

4.4 Performance Metrics

Table 1: Performance Metrics for Team and Player Models

Model	Accuracy	Precision (Class 1)	Recall (Class 1)	AUC
Team (High Performance)	0.96	1.00	0.69	0.83
Player (High Goals)	0.88	1.00	0.79	∼0.80

4.5 Discussion

The team model achieves high accuracy (0.96) but lower recall (0.69) for High Performance due to imbalance (3 High vs. 23 Low teams). The player model (0.88 accuracy, 0.79 recall) shows overprediction in some cases (e.g., Angola: actual=0, predicted=1). Key features (`avg_goals_scored`, `num_players_scored`) align with soccer intuition.

4.6 Challenges and Limitations

- **Data Imbalance**: Few High Performance (3/26) and High Goals teams skew predictions, reducing recall.
- **Shallow Trees**: Limited depth (e.g., 2 for team model) may miss complex patterns.
- **Overprediction**: Player model overpredicts High Goals, suggesting overfitting.

Future solutions include SMOTE for balancing, deeper trees, or Random Forests.

5 Time Complexity

- **MapReduce**: $O(n/p)$ for feature extraction, where n is data size and p is the number of processors.
- **Decision Tree Training**: $O(m \cdot n \cdot \log n)$, where m is the number of features and n is the number of samples.
- **Prediction**: $O(d)$, where d is tree depth (e.g., 2 for team model).

MapReduce ensures scalability, while shallow trees enable fast predictions.

6 Conclusion

This project developed a scalable MapReduce-based decision tree system for FIFA World Cup performance prediction, achieving 96% and 88% accuracy for team and player models, respectively. Despite strong AUCs ($\sim$0.83), imbalance limited recall. Future work includes multi-node Hadoop deployment and ensemble models. The project showcases the synergy of big data and machine learning in sports analytics.

7 Contributions

Aqsat Asif and Harsh Patel equally contributed to project design, MapReduce scripts, decision tree models, and visualizations.

8 References

1. Witten, I. H., Frank, E., & Hall, M. A. (2016). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.
2. Hadoop Documentation. (2023). Hadoop Streaming. Available at: `https://hadoop.apache.org/docs/r3.3.4/`
3. scikit-learn Documentation. (2023). DecisionTreeClassifier, roc_curve, auc. Available at: `https://scikit-learn.org/stable/`
4. Provost, F., & Fawcett, T. (2013). *Data Science for Business*. O'Reilly Media.