Supplementary Material: Rule-Based Classification and Model Visualizations

Mehmet Gür

1. Rule-Based Classification Functions

The following pseudocode illustrates the rule-based scoring functions used to assign player roles based on FIFA skill attributes.

1.1 General Position Classification (4 Roles)

```
def guess_position(row):
    gk = (
        0.45 * row['gk_reflexes'] +
        0.25 * row['gk_positioning'] +
        0.15 * row['gk_diving'] +
        0.10 * row['gk_handling'] +
        0.05 * row['gk_kicking']
    )

    defense = (
        0.30 * row['marking'] +
        0.20 * row['standing_tackle'] +
        0.15 * row['sliding_tackle'] +
        0.15 * row['interceptions'] +
        0.10 * row['strength'] +
        0.10 * row['aggression']
    )

    midfield = (
        0.25 * row['vision'] +
        0.20 * row['short_passing'] +
        0.15 * row['long_passing'] +
        0.15 * row['ball_control'] +
        0.15 * row['dribbling'] +
        0.10 * row['agility']
    )

    forward = (
        0.30 * row['finishing'] +
        0.20 * row['volleys'] +
        0.20 * row['positioning'] +
```

```
        0.15 * row['shot_power'] +
        0.15 * row['heading_accuracy']
    )

    return np.argmax([gk, defense, midfield, forward])
```

1.2 Detailed Position Classification (7 Roles)

```
def guess_detailed_position(row):
    scores = {}

    scores['GK'] = (
        0.45 * row['gk_reflexes'] +
        0.25 * row['gk_positioning'] +
        0.15 * row['gk_diving'] +
        0.10 * row['gk_handling'] +
        0.05 * row['gk_kicking']
    )

    scores['Center Defense'] = (
        0.25 * row['marking'] +
        0.20 * row['standing_tackle'] +
        0.20 * row['sliding_tackle'] +
        0.15 * row['interceptions'] +
        0.10 * row['strength'] +
        0.10 * row ['heading_accuracy']
    )

    scores['Side Defense'] = (
        0.25 * row['crossing'] +
        0.20 * row['acceleration'] +
        0.20 * row['sprint_speed'] +
        0.15 * row['agility'] +
        0.20 * row['standing_tackle']
    )

    scores['Center Midfield'] = (
        0.25 * row['vision'] +
        0.20 * row['short_passing'] +
        0.15 * row['long_passing'] +
        0.15 * row['interceptions'] +
        0.15 * row['standing_tackle'] +
        0.10 * row['aggression']
    )

    scores['Side Midfield'] = (
        0.20 * row['crossing'] +
        0.20 * row['sprint_speed'] +
```

```
        0.15 * row['agility'] +
        0.15 * row['dribbling'] +
        0.10 * row['curve'] +
        0.10 * row['acceleration'] +
        0.10 * row['shot_power']
    )

    scores['Center Forward'] = (
        0.30 * row['finishing'] +
        0.20 * row['positioning'] +
        0.15 * row['volleys'] +
        0.10 * row['shot_power'] +
        0.15 * row['heading_accuracy'] +
        0.10 * row['strength']
    )

    scores['Side Forward'] = (
        0.25 * row['acceleration'] +
        0.20 * row['dribbling'] +
        0.20 * row['agility'] +
        0.15 * row['crossing'] +
        0.10 * row['finishing'] +
        0.10 * row['curve']
    )

    return max(scores, key=scores.get)
```

(The full weighted scoring logic is included in the main paper's methods section.)

2. Confusion Matrices

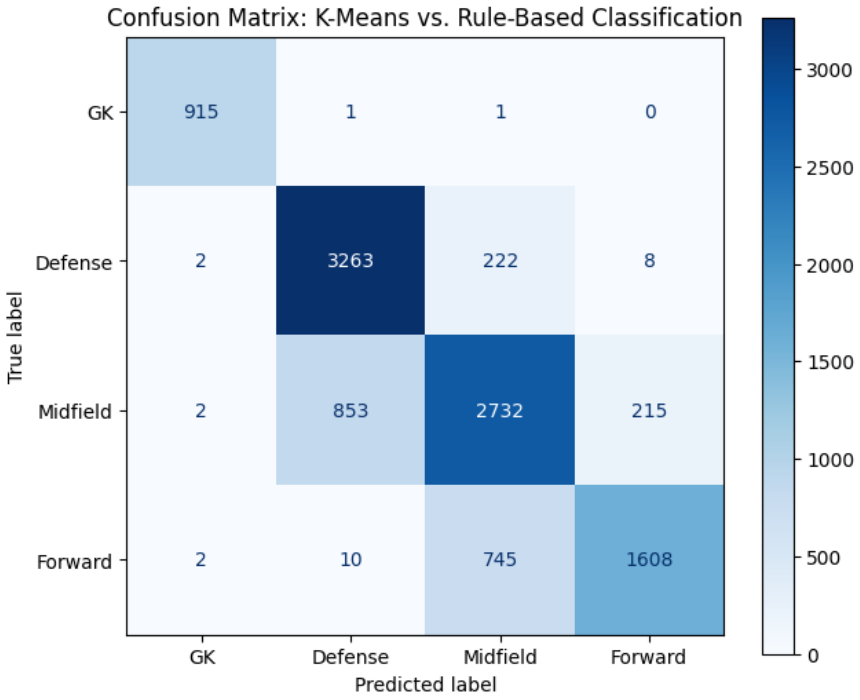

Figure 1: Confusion Matrix: K-Means vs. Rule-Based (4 Roles)

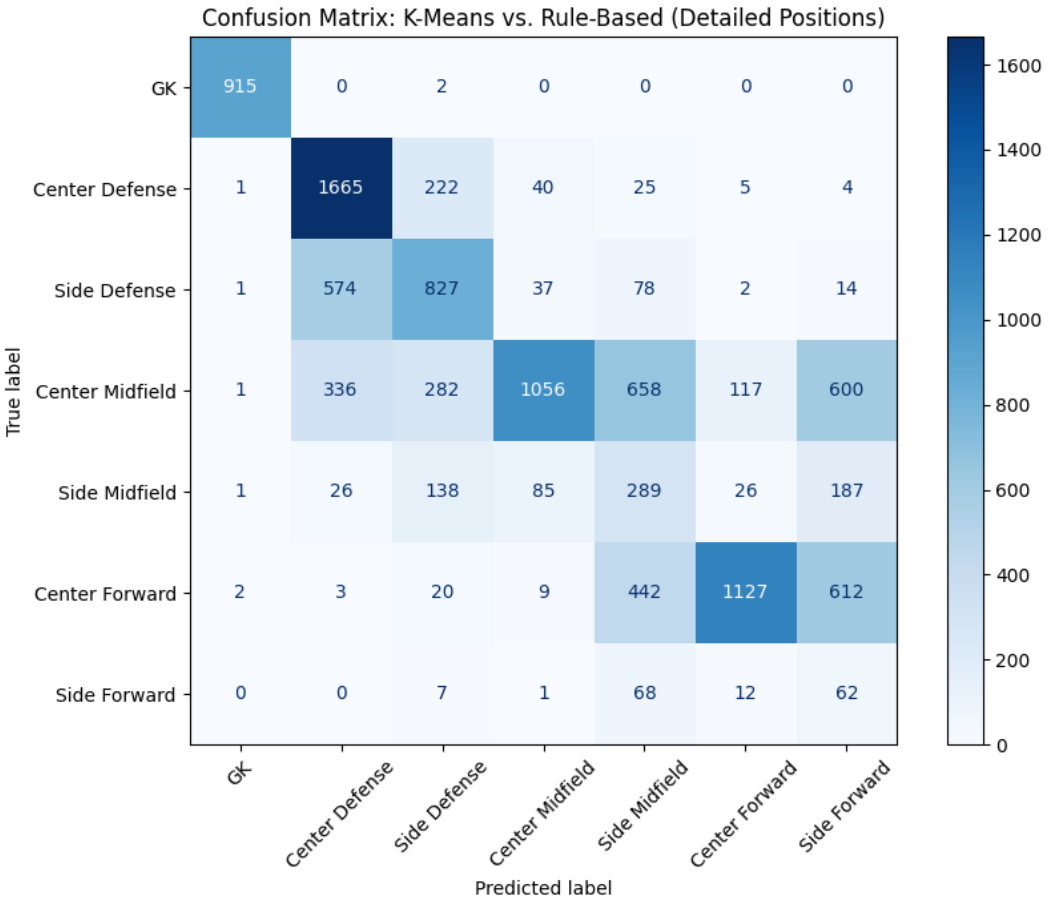

Figure 2: Confusion Matrix: K-Means vs. Rule-Based (7 Roles)

3. RMSE and R^2 Comparisons by Position

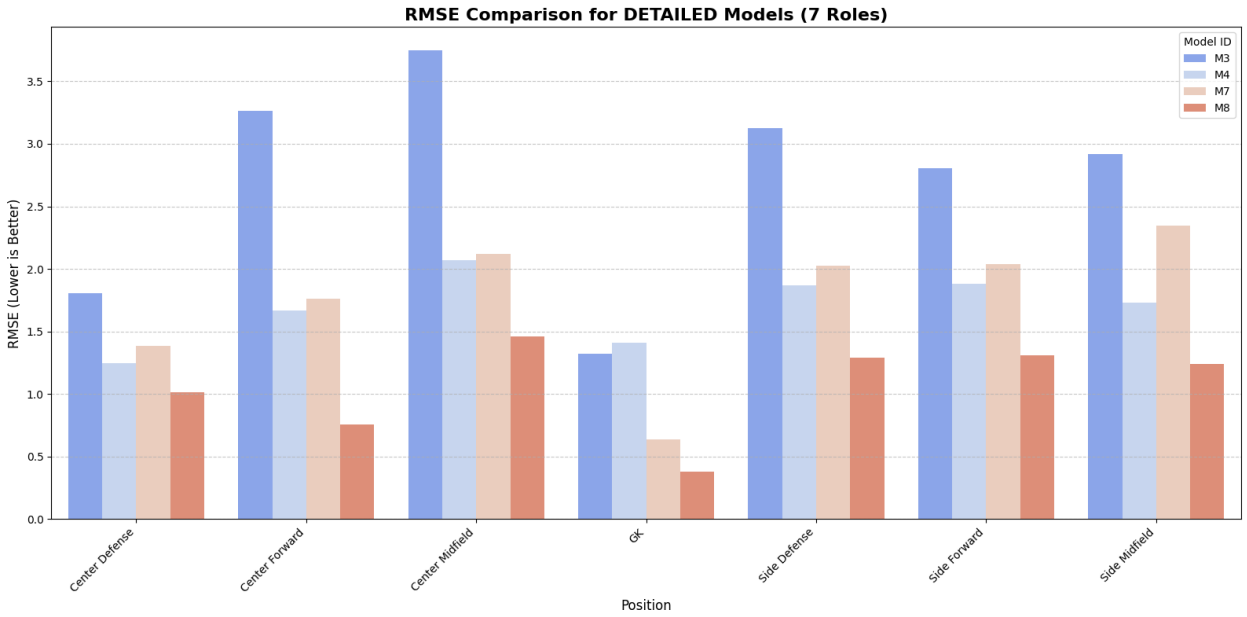

Figure 3: RMSE Comparison by Role (7-Role Models)

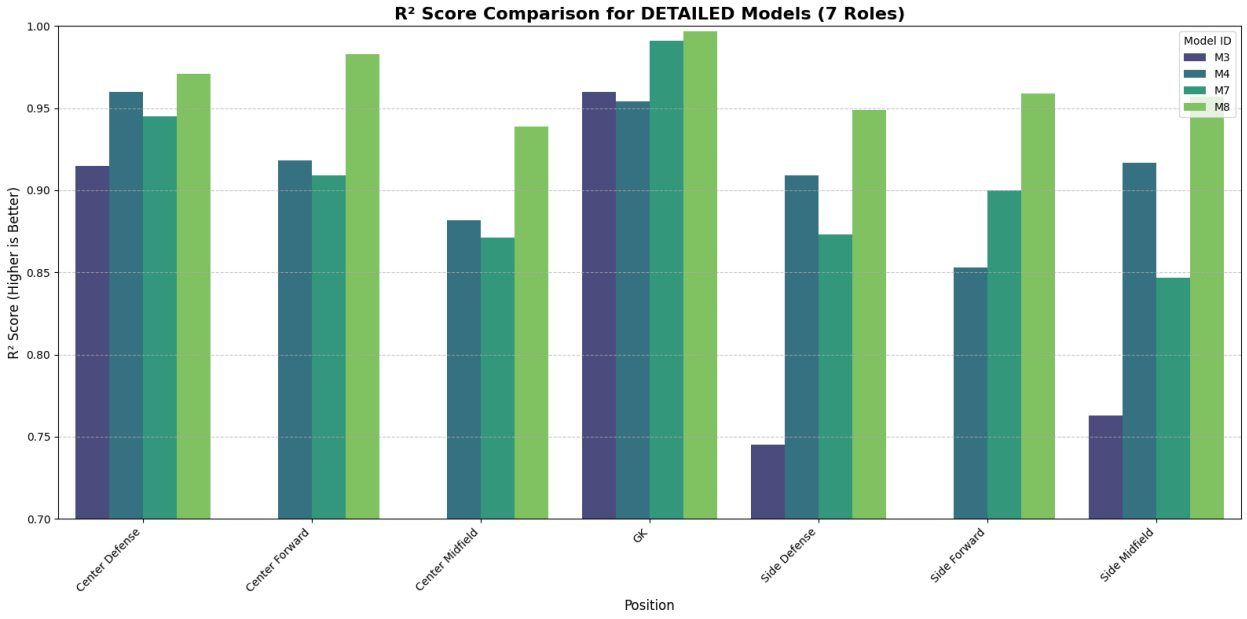

Figure 4: R^2 Comparison by Role (7-Role Models)

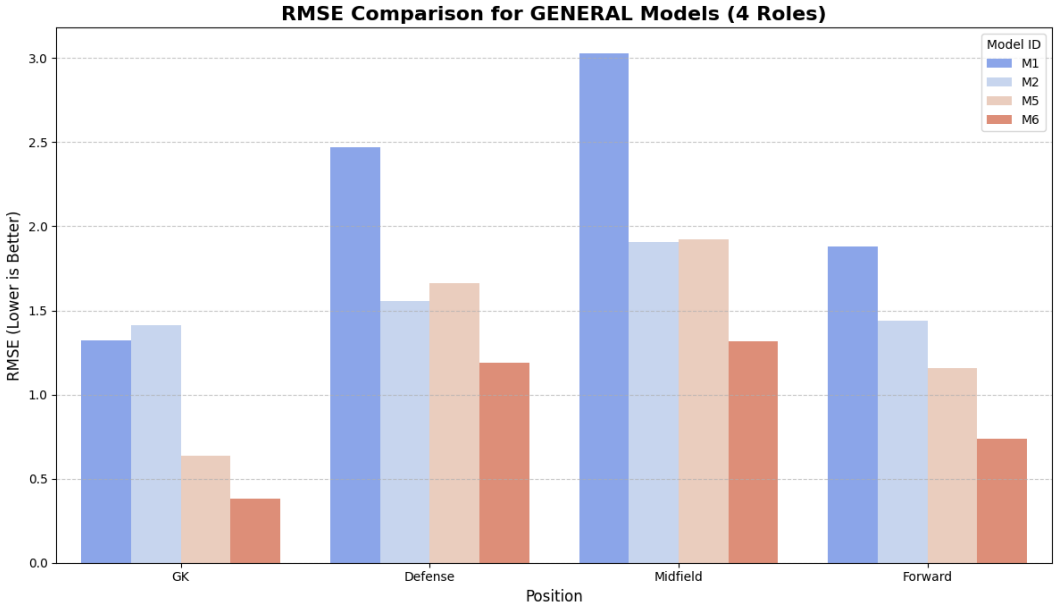

Figure 5: RMSE Comparison by Role (4-Role Models)

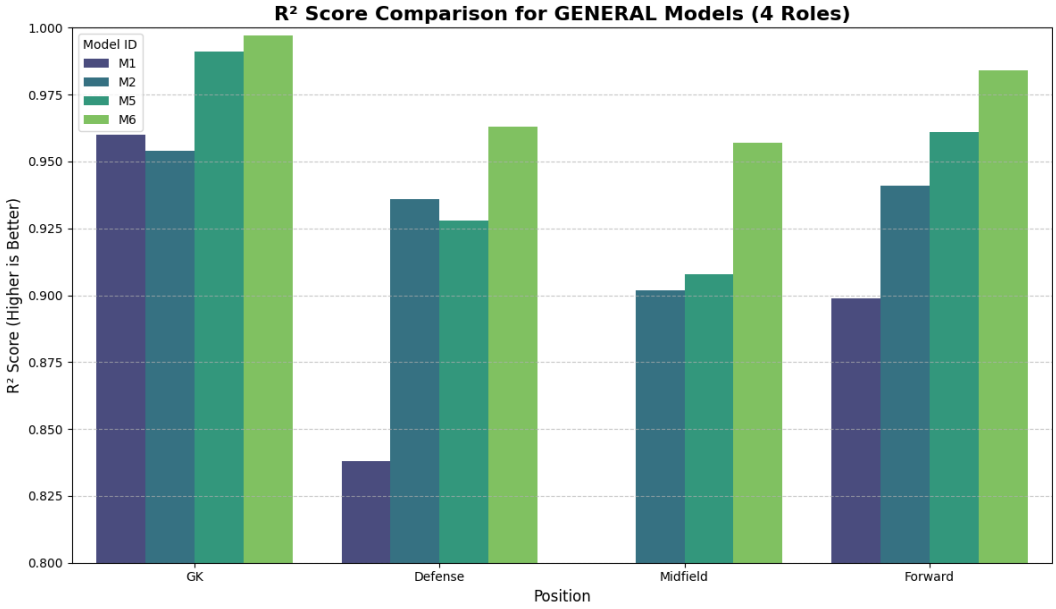

Figure 6: R^2 Comparison by Role (4-Role Models)

4. Feature Importance Visualizations (M6 - Ridge)

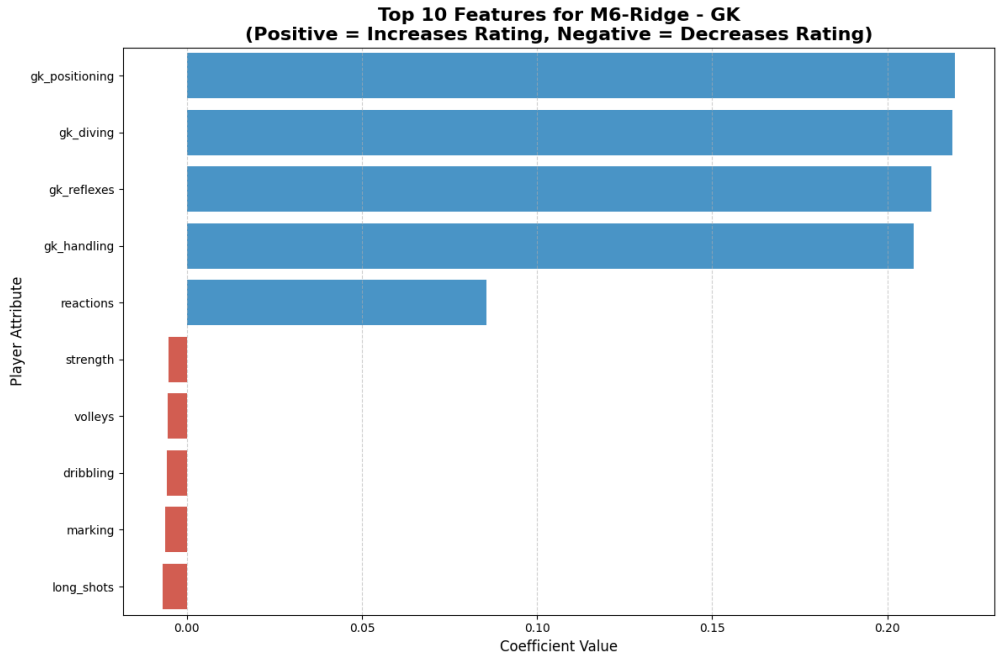

Figure 7: Top Features for Goalkeepers

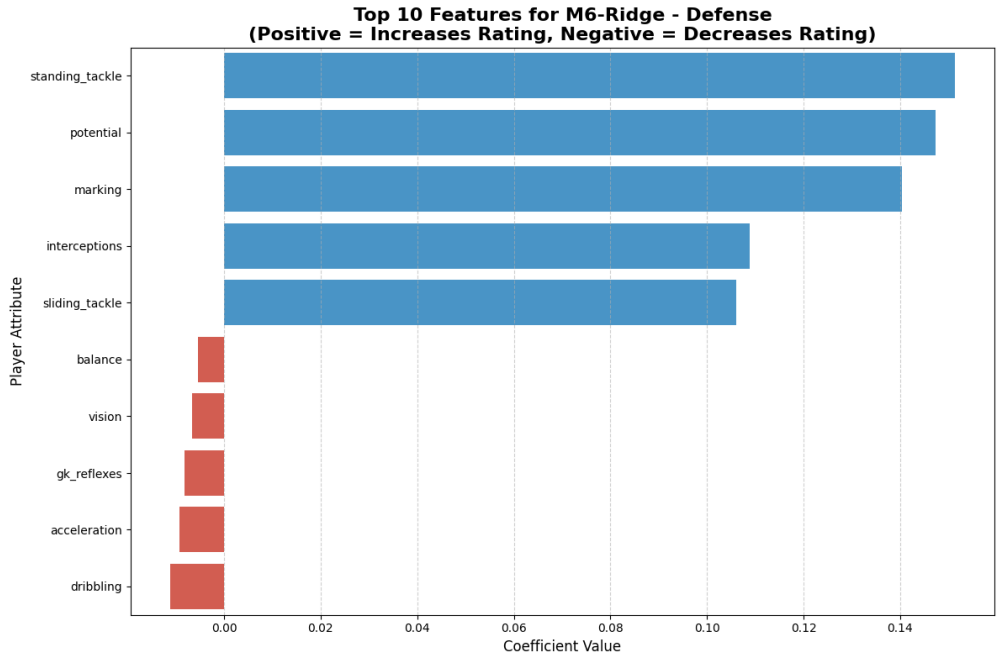

Figure 8: Top Features for Defenders

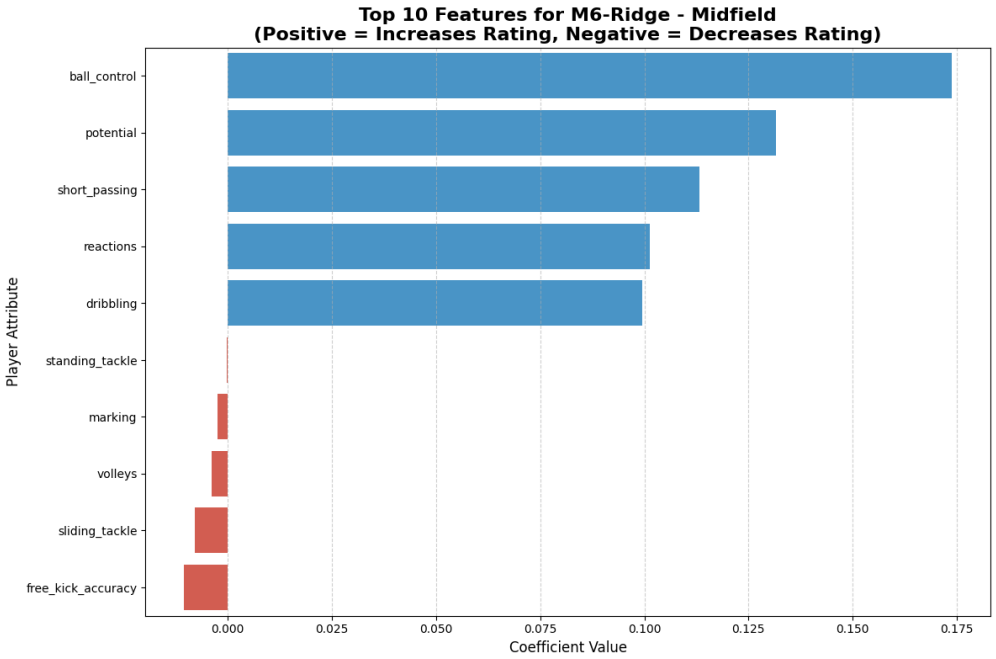

Figure 9: Top Features for Midfielders

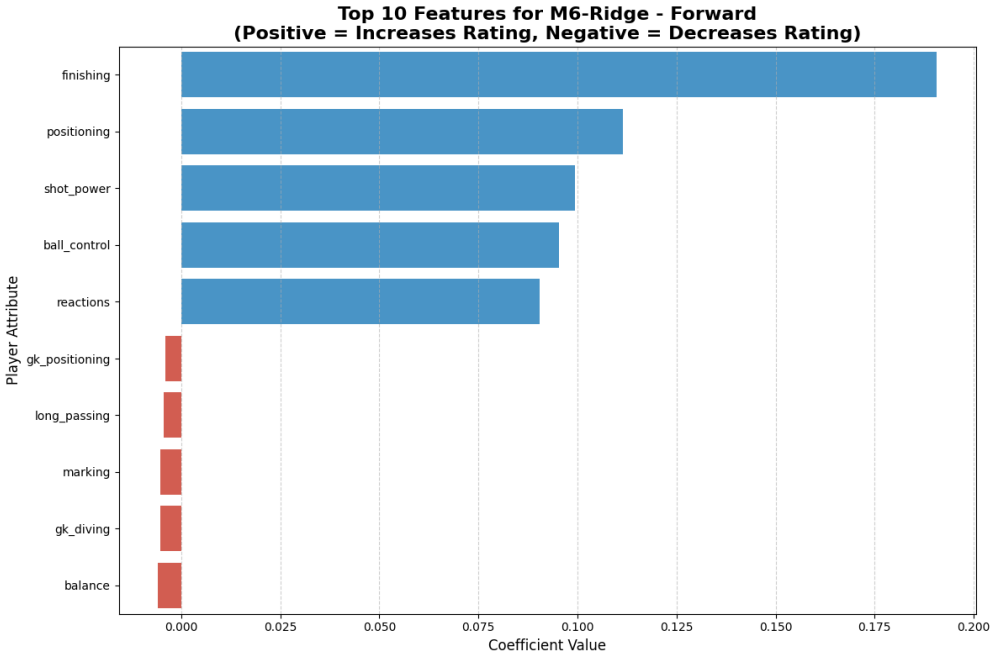

Figure 10: Top Features for Forwards