1. If You Are Predicting the Winner (home_win):

Drop visitor_points and home_points:

- These columns directly determine the winner (home_win) and would introduce data leakage into your model.
- Data leakage occurs when information that would not be available at prediction time is included in the training data.

Example:

python

Copy code

```
X = encoded_df.drop(columns=['home_win', 'visitor_points', 'home_points'])

y = encoded_df['home_win']
```

---

2. If You Are Predicting the Total Points (visitor_points + home_points):

Combine visitor_points and home_points:

- Use the total points as your target variable (y).
- Drop both visitor_points and home_points from the features (X).

Example:

python

Copy code

```
encoded_df['total_points'] = encoded_df['visitor_points'] + encoded_df['home_points']

X = encoded_df.drop(columns=['visitor_points', 'home_points', 'total_points'])

y = encoded_df['total_points']
```

---

3. If You Are Predicting Team Scores Individually:

Split the Target Variable (y) into Two Columns:

- For a multi-output regression model, you can predict both visitor_points and home_points simultaneously.
- Keep all other columns (e.g., one-hot-encoded team names) as features.

Example:

python

Copy code

```
X = encoded_df.drop(columns=['visitor_points', 'home_points'])

y = encoded_df[['visitor_points', 'home_points']]
```

Alternatively, Predict One Team's Score at a Time:

- If you want to predict only one team's score, use either visitor_points or home_points as the target variable.

---

4. If You Want to Include Team Points as Features:

Use Aggregated Statistics Instead of Raw Points:

- Historical averages for points scored/allowed by each team are more predictive than the actual game points.
- Compute statistics like:
    - **Average Points Scored**: mean(visitor_points) and mean(home_points)
    - **Points Allowed**: Opponent's average points in prior games.

Example:

python

Copy code

```
encoded_df['visitor_avg_points'] =
encoded_df.groupby('visitor_team')['visitor_points'].transform('mean')

encoded_df['home_avg_points'] =
encoded_df.groupby('home_team')['home_points'].transform('mean')


X = encoded_df.drop(columns=['visitor_points', 'home_points', 'home_win'])

y = encoded_df['home_win']  # For winner prediction
```

Final Checklist Before Modeling:

1. **Avoid Data Leakage**:
    - Do not include visitor_points or home_points directly if they correlate strongly with the target (home_win or total points).
2. **Scale Features**:

- Normalize or standardize numerical features like aggregated statistics to avoid dominance by large-scale features.

3. **Include Meaningful Features**:
    - Enrich the dataset with team performance statistics, win/loss streaks, and other context.
4. **Drop Redundant Columns**:
    - Columns like home_win should only be included as the target (y) and excluded from X.

With these steps, your machine learning process will have the right input data for accurate predictions.