**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.