# RANKING

## DATA PREPROCESSING

In [1]:
```
import pandas as pd  # Import the pandas library for data manipulation

# Load the dataset containing game statistics
df = pd.read_csv(r"games_2022.csv")

# Drop unnecessary columns to reduce data size and focus on relevant statistics
df.drop(columns=['travel_dist', 'home_away_NS', 'home_away', 'attendance', 'game_date',
                 'tz_dif_H_E', 'prev_game_dist', 'rest_days', 'OT_length_min_tot'],
        inplace=True, errors='ignore')

# Load the dataset containing team-region mapping
regions = pd.read_csv(r"E:\basketball_project\Team Region Groups.csv")

# Merge the main dataset with the region mapping based on the 'team' column
region_df = df.merge(regions, on="team", how="left")

# Fill missing region values with 'East' as the default region
region_df['region'] = region_df['region'].fillna('East')

# Calculate field goal percentage for 2-point shots
region_df['FGP_2'] = ((region_df['FGM_2'] / region_df['FGA_2']) * 100).replace([float('inf'), -float('inf')], 0

# Calculate field goal percentage for 3-point shots
region_df['FGP_3'] = ((region_df['FGM_3'] / region_df['FGA_3']) * 100).replace([float('inf'), -float('inf')], 0

# Calculate free throw percentage
region_df['FT_Percentage'] = ((region_df['FTM'] / region_df['FTA']) * 100).replace([float('inf'), -float('inf')

# Compute total rebounds by summing offensive and defensive rebounds
region_df['Total_Rebounds'] = region_df['DREB'] + region_df['OREB']

# Assign game result: 'True' for win, 'False' for loss
region_df['result'] = region_df.apply(lambda row: True if row['team_score'] > row['opponent_team_score']
                                      else False, axis=1)
region_df['result'] = region_df['result'].astype(bool)  # Ensure the result column is boolean

# Remove columns that are no longer needed after calculations
region_df.drop(columns=['FGA_2', 'FGA_3', 'FGM_2', 'FGM_3', 'FTA', 'FTM',
                        'team_score', 'opponent_team_score', 'OREB', 'DREB',
                        'notD1_incomplete'], inplace=True, errors='ignore')

# Group data by 'team' and calculate mean values for statistical features
teamwise_stats = region_df.groupby('team').agg({
    'AST': 'mean',  # Average assists
    'BLK': 'mean',  # Average blocks
    'STL': 'mean',  # Average steals
    'TOV': 'mean',  # Average turnovers
    'TOV_team': 'mean',  # Average team turnovers
    'F_tech': 'mean',  # Average technical fouls
    'F_personal': 'mean',  # Average personal fouls
    'largest_lead': 'mean',  # Average largest lead
    'FGP_2': 'mean',  # Average 2-point field goal percentage
    'FGP_3': 'mean',  # Average 3-point field goal percentage
    'FT_Percentage': 'mean',  # Average free throw percentage
    'Total_Rebounds': 'mean',  # Average total rebounds
    'result': ['sum', 'count'],  # Total wins and total matches played
    'region': 'first'  # Keep the region name (assuming one region per team)
})

# Rename columns for clarity
teamwise_stats.columns = ['Avg_AST', 'Avg_BLK', 'Avg_STL', 'Avg_TOV', 'Avg_TOV_team',
                          'Avg_F_tech', 'Avg_F_personal', 'Avg_largest_lead',
                          'Avg_FGP_2', 'Avg_FGP_3', 'Avg_FT_Percentage', 'Avg_Total_Rebounds',
                          'Total_Wins', 'Total_Matches', 'Region']

# Save the aggregated team statistics to a CSV file
teamwise_stats.to_csv('teamwise_stats.csv')

# Define regions for filtering
regions = ['East', 'West', 'North', 'South']

# Split the dataset by region and save separate files
for region in regions:
    # Filter data for the specific region
```

```
    region_df = teamwise_stats[teamwise_stats['Region'] == region]

    # Save the region-specific statistics to a CSV file
    filename = f'teamwise_stats_{region.lower()}.csv'
    region_df.to_csv(filename, index=True)

    # Print confirmation message
    print(f"Saved: {filename}")
```

```
Saved: teamwise_stats_east.csv
Saved: teamwise_stats_west.csv
Saved: teamwise_stats_north.csv
Saved: teamwise_stats_south.csv
```

# REGION-WISE RANKING

In [3]:

```
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# List of input files to process
files = [
    "teamwise_stats_north.csv",
    "teamwise_stats_south.csv",
    "teamwise_stats_east.csv",
    "teamwise_stats_west.csv"
]

# Loop through each file to process the data
for file in files:
    # Load the data for the current file
    df = pd.read_csv(file)

    # Create a more balanced ranking score
    df['Ranking_Score'] = df['Total_Wins'] * (df['Total_Wins'] / (df['Total_Matches'] + 1))

    # Select features
    features = ['Avg_AST', 'Avg_BLK', 'Avg_STL', 'Avg_TOV', 'Avg_TOV_team',
                'Avg_F_tech', 'Avg_F_personal', 'Avg_largest_lead',
                'Avg_FGP_2', 'Avg_FGP_3', 'Avg_FT_Percentage', 'Avg_Total_Rebounds', 'Total_Matches']

    X = df[features]
    y = df['Ranking_Score']

    # Train-Test Split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Scale Data
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    # Train XGBoost Model
    model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100, learning_rate=0.1)
    model.fit(X_train_scaled, y_train)

    # Predict ranking scores for the entire dataset
    df['Predicted_Ranking_Score'] = model.predict(scaler.transform(X))

    # Rank teams based on predicted scores
    df = df.sort_values(by='Predicted_Ranking_Score', ascending=False)
    df['Rank'] = range(1, len(df) + 1)

    # Save results for each region
    output_file = file.replace(".csv", "_ranking.csv")
    df.to_csv(output_file, index=False)
    print(f"Team ranking saved as {output_file}")
```

```
Team ranking saved as teamwise_stats_north_ranking.csv
Team ranking saved as teamwise_stats_south_ranking.csv
Team ranking saved as teamwise_stats_east_ranking.csv
Team ranking saved as teamwise_stats_west_ranking.csv
```

# EXPLANATION OF ABOVE CODE

## RANKING

• Data Cleaning and Transformation: o The initial dataset (games_2022.csv) was loaded using pandas. o Irrelevant columns such as

travel_dist, home_away_NS, home_away, attendance, game_date, tz_dif_H_E, prev_game_dist, rest_days, and OT_length_min_tot were dropped to streamline the data and focus on relevant features. • Feature Engineering: o A region column was added to the main dataframe by merging it with the Team Region Groups.csv file based on the team name. Missing region values were filled with "East". o Several new features were engineered to provide additional insights: 👇 FGP_2: Two-point field goal percentage was calculated. 👇 FGP_3: Three-point field goal percentage was calculated. 👇 FT_Percentage: Free throw percentage was calculated. 👇 Total_Rebounds: Total rebounds were calculated by summing defensive and offensive rebounds. 👇 The code handles potential division by zero errors by replacing infinite values with 0 and filling any remaining NaN values with 0. The percentage columns are rounded to three decimal places. o A Boolean result column was created to indicate whether a team won (True) or lost (False) a game, based on comparing t0eam_score and opponent_team_score. • Data Reduction: o Original columns used for percentage calculations and the result column, such as FGA_2, FGA_3, FGM_2, FGM_3, FTA, FTM, team_score, opponent_team_score, OREB, and DREB, were dropped to reduce dimensionality and avoid redundancy. o The notD1_incomplete column was dropped. • Tools and Techniques: o Pandas: This Python library was used for data loading, merging, cleaning, transformation, and feature engineering. o Statistical Methods: Basic arithmetic calculations were used to create new features such as field goal percentages and total rebounds. Conditional logic was applied to determine game results. • Saving Refined Data: o The final refined dataset was saved to a new CSV file (refined_data.csv) for subsequent analysis and modelling.

## Why Use XGBoost for Ranking?

1. Learning Complex Relationships

o Traditional ranking formulas assign fixed weights, but team performance is influenced by complex interactions between stats (assists, steals, rebounds, shooting accuracy, etc.). o XGBoost learns these non-linear patterns automatically. 2. Data-Driven Ranking Instead of Arbitrary Weights o Many traditional ranking methods rely on manually assigned weights for different features. o Instead, we let XGBoost determine the best weights based on historical performance. 3. Handling Multiple Factors Together o Factors like turnovers, fouls, and shooting percentages have different impacts on ranking. o XGBoost optimally balances these factors. 4. Generalizability & Adaptability o If new data is added (e.g., additional stats, new teams, or new matches), XGBoost automatically adapts without needing to tweak the formula. 5. Proven Accuracy o You obtained $R^2$ Score: 0.972, which means the model explains 97.2% of ranking variations, proving its effectiveness. o A traditional formula may not achieve such high accuracy. _____

## Explain Train-Test Split?

Your teacher may ask: **?** “Why are we doing a train-test split if we don’t have actual rankings?” ✅ Answer: The train-test split is used to train XGBoost to predict the Ranking Score based on past match statistics. Even though we don’t have predefined rankings, we assume that "Total Wins" is a strong ranking signal and use it to train the model.

## TO FETCH TOP 16 TEAMS

In [2]:
```
import pandas as pd

# Load the CSV file
df = pd.read_csv("teamwise_stats_north_ranking.csv")

# Select the top 16 teams based on Rank
top_16_teams = df.nsmallest(16, 'Rank')['team']

# Print the result in the desired format
print("---------NORTH REGION TOP 16 TEAMS----------------")
for i, team in enumerate(top_16_teams, start=1):
    print(f"{i}-{team}")
```

```
---------NORTH REGION TOP 16 TEAMS----------------
1-south_carolina_gamecocks
2-florida_gulf_coast_eagles
3-stephen_f_austin_ladyjacks
4-ucf_knights
5-middle_tennessee_blue_raiders
6-charleston_cougars
7-tennessee_lady_volunteers
8-belmont_bruins
9-jacksonville_state_gamecocks
10-jackson_state_lady_tigers
11-charlotte_49ers
12-ole_miss_rebels
13-lsu_tigers
14-troy_trojans
15-georgia_tech_yellow_jackets
16-florida_gators
```

In [3]:
```
import pandas as pd

# Load the CSV file
df = pd.read_csv("teamwise_stats_south_ranking.csv")
```

```
# Select the top 16 teams based on Rank
top_16_teams = df.nsmallest(16, 'Rank')['team']

# Print the result in the desired format
print("---------SOUTH REGION TOP 16 TEAMS----------------")
for i, team in enumerate(top_16_teams, start=1):
    print(f"{i}-{team}")
```

```
---------SOUTH REGION TOP 16 TEAMS----------------
1-louisville_cardinals
2-iowa_state_cyclones
3-iu_indianapolis_jaguars
4-virginia_tech_hokies
5-missouri_state_lady_bears
6-dayton_flyers
7-youngstown_state_penguins
8-iowa_hawkeyes
9-indiana_hoosiers
10-toledo_rockets
11-kansas_city_roos
12-cleveland_state_vikings
13-drake_bulldogs
14-notre_dame_fighting_irish
15-northern_kentucky_norse
16-depaul_blue_demons
```

In [4]:
```
import pandas as pd

# Load the CSV file
df = pd.read_csv("teamwise_stats_east_ranking.csv")

# Select the top 16 teams based on Rank
top_16_teams = df.nsmallest(16, 'Rank')['team']

# Print the result in the desired format
print("---------EAST REGION TOP 16 TEAMS----------------")
for i, team in enumerate(top_16_teams, start=1):
    print(f"{i}-{team}")
```

```
---------EAST REGION TOP 16 TEAMS----------------
1-nc_state_wolfpack
2-liberty_flames
3-drexel_dragons
4-massachusetts_minutewomen
5-uconn_huskies
6-lehigh_mountain_hawks
7-campbell_fighting_camels
8-princeton_tigers
9-buffalo_bulls
10-towson_tigers
11-delaware_blue_hens
12-rhode_island_rams
13-columbia_lions
14-stony_brook_seawolves
15-american_university_eagles
16-bucknell_bison
```

In [5]:
```
import pandas as pd

# Load the CSV file
df = pd.read_csv("teamwise_stats_west_ranking.csv")

# Select the top 16 teams based on Rank
top_16_teams = df.nsmallest(16, 'Rank')['team']

# Print the result in the desired format
print("---------WEST REGION TOP 16 TEAMS----------------")
for i, team in enumerate(top_16_teams, start=1):
    print(f"{i}-{team}")
```

```
---------WEST REGION TOP 16 TEAMS----------------
1-stanford_cardinal
2-baylor_bears
3-byu_cougars
4-texas_longhorns
5-gonzaga_bulldogs
6-unlv_lady_rebels
7-south_dakota_coyotes
8-oklahoma_sooners
9-nebraska_cornhuskers
10-south_dakota_state_jackrabbits
11-california_baptist_lancers
12-colorado_buffaloes
13-san_diego_toreros
14-grand_canyon_lopes
15-montana_state_bobcats
16-arizona_wildcats
```

In [ ]: