```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor

df = pd.read_csv("EPL.csv")

def compute_team_stats(df):
    team_stats = {}

    for team in pd.unique(df[['HomeTeam',
'AwayTeam']].values.ravel('K')):
        home_matches = df[df['HomeTeam'] == team]
        away_matches = df[df['AwayTeam'] == team]

        home_wins = (home_matches['FullTimeResult'] == 'H').sum()
        away_wins = (away_matches['FullTimeResult'] == 'A').sum()
        home_games = len(home_matches)
        away_games = len(away_matches)

        home_win_pct = home_wins / home_games if home_games > 0 else 0
        away_win_pct = away_wins / away_games if away_games > 0 else 0

        home_avg_goals = home_matches['FullTimeHomeTeamGoals'].mean()
if home_games > 0 else 0
        away_avg_goals = away_matches['FullTimeAwayTeamGoals'].mean()
if away_games > 0 else 0
        home_avg_shots = home_matches['HomeTeamShots'].mean() if
home_games > 0 else 0
        away_avg_shots = away_matches['AwayTeamShots'].mean() if
away_games > 0 else 0

        home_avg_shots_on_target =
home_matches['HomeTeamShotsOnTarget'].mean() if home_games > 0 else 0
        away_avg_shots_on_target =
away_matches['AwayTeamShotsOnTarget'].mean() if away_games > 0 else 0
        home_avg_fouls = home_matches['HomeTeamFouls'].mean() if
home_games > 0 else 0
        away_avg_fouls = away_matches['AwayTeamFouls'].mean() if
away_games > 0 else 0

        team_stats[team] = {
            'home_win_pct': home_win_pct,
            'away_win_pct': away_win_pct,
            'home_avg_goals': home_avg_goals,
            'away_avg_goals': away_avg_goals,
            'home_avg_shots': home_avg_shots,
            'away_avg_shots': away_avg_shots,
            'home_avg_shots_on_target': home_avg_shots_on_target,
            'away_avg_shots_on_target': away_avg_shots_on_target,
            'home_avg_fouls': home_avg_fouls,
            'away_avg_fouls': away_avg_fouls
        }

    return team_stats

team_stats = compute_team_stats(df)

features = []
goal_labels = []
shot_labels = []
shot_on_target_labels = []
foul_labels = []

for _, row in df.iterrows():
    home_team = row['HomeTeam']
    away_team = row['AwayTeam']

    if home_team in team_stats and away_team in team_stats:
        home = team_stats[home_team]
        away = team_stats[away_team]

        features.append([
            home['home_win_pct'], away['away_win_pct'],
            home['home_avg_goals'], away['away_avg_goals'],
            home['home_avg_shots'], away['away_avg_shots'],
            home['home_avg_shots_on_target'],
away['away_avg_shots_on_target'],
            home['home_avg_fouls'], away['away_avg_fouls']
        ])

        goal_labels.append([row['FullTimeHomeTeamGoals'],
row['FullTimeAwayTeamGoals']])
        shot_labels.append([row['HomeTeamShots'],
row['AwayTeamShots']])
        shot_on_target_labels.append([row['HomeTeamShotsOnTarget'],
row['AwayTeamShotsOnTarget']])
        foul_labels.append([row['HomeTeamFouls'],
row['AwayTeamFouls']])

X = np.array(features)
y_goals = np.array(goal_labels)
y_shots = np.array(shot_labels)
y_shots_on_target = np.array(shot_on_target_labels)
y_fouls = np.array(foul_labels)

X_train_g, X_test_g, y_train_g, y_test_g = train_test_split(X,
y_goals, test_size=0.1, random_state=42)
X_train_s, X_test_s, y_train_s, y_test_s = train_test_split(X,
y_shots, test_size=0.1, random_state=42)
```

```
X_train_st, X_test_st, y_train_st, y_test_st = train_test_split(X,
y_shots_on_target, test_size=0.1, random_state=42)
X_train_f, X_test_f, y_train_f, y_test_f = train_test_split(X,
y_fouls, test_size=0.1, random_state=42)

goal_regressor = DecisionTreeRegressor(random_state=42)
goal_regressor.fit(X_train_g, y_train_g)

shot_regressor = DecisionTreeRegressor(random_state=42)
shot_regressor.fit(X_train_s, y_train_s)

shot_on_target_regressor = DecisionTreeRegressor(random_state=42)
shot_on_target_regressor.fit(X_train_st, y_train_st)

foul_regressor = DecisionTreeRegressor(random_state=42)
foul_regressor.fit(X_train_f, y_train_f)
```

```
DecisionTreeRegressor(random_state=42)
```

```
def predict_match(home_team, away_team):
    if home_team not in team_stats or away_team not in team_stats:
        return "Invalid teams!"

    home = team_stats[home_team]
    away = team_stats[away_team]

    input_features = np.array([[
        home['home_win_pct'], away['away_win_pct'],
        home['home_avg_goals'], away['away_avg_goals'],
        home['home_avg_shots'], away['away_avg_shots'],
        home['home_avg_shots_on_target'],
away['away_avg_shots_on_target'],
        home['home_avg_fouls'], away['away_avg_fouls']
    ]]).reshape(1, -1)

    goal_pred = goal_regressor.predict(input_features)[0]
    shot_pred = shot_regressor.predict(input_features)[0]
    shot_on_target_pred =
shot_on_target_regressor.predict(input_features)[0]
    foul_pred = foul_regressor.predict(input_features)[0]

    home_goals, away_goals = int(round(goal_pred[0])),
int(round(goal_pred[1]))
    home_shots, away_shots = int(round(shot_pred[0])),
int(round(shot_pred[1]))
    home_shots_on_target, away_shots_on_target =
int(round(shot_on_target_pred[0])), int(round(shot_on_target_pred[1]))
    home_fouls, away_fouls = int(round(foul_pred[0])),
int(round(foul_pred[1]))

    if home_goals > away_goals:
        result_pred = "H"
    elif away_goals > home_goals:
        result_pred = "A"
    else:
        result_pred = "D"

    return f"""
    Predicted Outcome: {result_pred}
    Expected Full-Time Goals: {home_team} {home_goals} - {away_goals}
{away_team}
    Expected Shots: {home_team} {home_shots} - {away_shots}
{away_team}
    Expected Shots on Target: {home_team} {home_shots_on_target} -
{away_shots_on_target} {away_team}
    Expected Fouls: {home_team} {home_fouls} - {away_fouls}
{away_team}
    """


home_team = "Liverpool"
away_team = "Man City"
print(predict_match(home_team, away_team))
```

```
    Predicted Outcome: H
    Expected Full-Time Goals: Liverpool 2 - 1 Man City
    Expected Shots: Liverpool 14 - 13 Man City
    Expected Shots on Target: Liverpool 5 - 4 Man City
    Expected Fouls: Liverpool 9 - 8 Man City
```