# Project 2

By: David Hoffman and Kyle Kolodziej

## 1: Getting Started

- Import libraries
- Load original Data (which ever one you chose from the provided list) into a data frame.
- Load your additional data set(s) into a data frame.
- In a markdown cell, provide a brief description of your the data sets you've chosen to work with.
- Develop a list of 3 - 4 questions that you hope to be able to answer after the exploration of the data and write them in this section.

```
import os

os.environ['KAGGLE_USERNAME'] = "davidhoffman11" # username from the json file
os.environ['KAGGLE_KEY'] = "43c89d1d4b7f8d60a63d6d531fc0b018" # key from the json file

!kaggle datasets download -d sanjeetsinghnaik/football-data-top-5-leagues
```

```
Downloading football-data-top-5-leagues.zip to /content
  0% 0.00/1.01M [00:00<?, ?B/s]
100% 1.01M/1.01M [00:00<00:00, 41.1MB/s]
```

```
# importing required modules
import numpy as np
import pandas as pd
from zipfile import ZipFile
import os
```

```
# specifying the zip file name
file_name = "football-data-top-5-leagues.zip"

# opening the zip file in READ mode
with ZipFile(file_name, 'r') as zip:
    # extracting all the files
    print('Extracting all the files now...')
    zip.extractall('./data')
    print('Done!')
```

```
Extracting all the files now...
Done!
```

```
# read in combined dataset into a dataframe
df = pd.read_csv("/content/data/combined_data.csv")

df.head(5)
```

The first of our datasets provides various game data from each of the top 5 soccer leagues in the world: The Premier League, Ligue, The Budesliga, Seria, and La Liga. This dataset contains data from games from 2014-2020 and includes many different game statistics such as each team's rating, the match excitement, team posession percentages, shots on goal, etc. To get this dataset into a pandas dataframe, we first had to download the dataset from kaggle. After this, the data was downloaded into our local environment in the form of a .zip file with several different files within it. We then exported all of the individual .csv files from the original .zip file and loaded the combined .csv into our original dataframe.

```
!kaggle datasets download -d thegreatcoder/points-table-of-5-leagues-in-football-20142018
```

```
Downloading points-table-of-5-leagues-in-football-20142018.zip to /content
  0% 0.00/17.8k [00:00<?, ?B/s]
100% 17.8k/17.8k [00:00<00:00, 4.41MB/s]
```

```
# specifying the zip file name
file_name = "points-table-of-5-leagues-in-football-20142018.zip"

# opening the zip file in READ mode
with ZipFile(file_name, 'r') as zip:
    # extracting all the files
    print('Extracting all the files now...')
    zip.extractall('./data')
    print('Done!')
```

```
Extracting all the files now...
Done!
```

```
# read in second dataset into a dataframe
df2 = pd.read_csv("/content/data/Football Data.csv")

df2.head(5)
```

Rather than showing game-to-game statistics, our second dataset gives year totals for each team in the five biggest soccer leagues in the world. This dataset contains information such as total matches, total wins, total losses, points scored, foul statistics, and different shooting statistics. Unlike the first dataset, this dataset only contains information for 2014-2018 missing data from the 2019 and 2020 season which is contained in the first dataset. We hope that this will not be a problem moving forward; however, if it does prove to be a problem we may need to inpute the values for the missing years or drop 2019 and 2020 from the first dataset altogether.

Questions we hope to answer after data exploration:

- What individual match statistics are most correlated with overall match excitment?
- What season statisitics are most correlated with season totals for match wins?
- What are the main statistical differences between teams at the top of the league and teams at the bottom?
- Which of the major leagues is the most exciting?

## 2. Data Inspection

Write some code to summarize the datasets. Think about the following questions:

- What type of data is each variable? (think like a data scientist here, not a computer scientist)
- What is the total size of the data sets?
- What time boundaries are there in the dataset? IOW, what time frame do they span?
- Are there any missing values in any of the variables?

Do this with Intentionality. Don't skimp.

```
# data types of first dataset attributes
df.dtypes
```

```
Unnamed: 0                         int64
Home Team                         object
Away Team                         object
Score                             object
Half Time Score                   object
Match Excitement                 float64
Home Team Rating                 float64
Away Team Rating                 float64
Home Team Possession %             int64
Away Team Possession %             int64
Home Team Off Target Shots       float64
Home Team On Target Shots        float64
Home Team Total Shots            float64
Home Team Blocked Shots          float64
Home Team Corners                float64
Home Team Throw Ins              float64
Home Team Pass Success %         float64
Home Team Aerials Won            float64
Home Team Clearances             float64
Home Team Fouls                  float64
Home Team Yellow Cards           float64
Home Team Second Yellow Cards    float64
Home Team Red Cards              float64
Away Team Off Target Shots       float64
Away Team On Target Shots        float64
Away Team Total Shots            float64
Away Team Blocked Shots          float64
Away Team Corners                float64
Away Team Throw Ins              float64
Away Team Pass Success %         float64
```

```
Away Team Aerials Won            float64
Away Team Clearances             float64
Away Team Fouls                  float64
Away Team Yellow Cards           float64
Away Team Second Yellow Cards    float64
Away Team Red Cards              float64
Home Team Goals Scored             int64
Away Team Goals Scored             int64
Home Team Goals Conceeded          int64
Away Team Goals Conceeded          int64
year                               int64
league                            object
dtype: object
```

With the first dataset, nearly all the features are comprised of numerical data as many of them are totals for each statistical category throughout the game. Despite this, there are also several features that currently contain categorical data the obvious ones being the league and home/away team names. In addition, the final score and halftime score are also currently listed as strings and could be interpretted as categorical data or numerical data depending on the context. The string representation of the halftime and final scores are caused because the feature contains both the away and home team's goal total seperated by a hyphen. It may be beneficial to divide this feature into two features (home score and away score), but I believe that the total score representation also has merit because it shows the entire picture indicating the closeness of the game.

```
# data types of second dataset attributes
df2.dtypes
```

```
League        object
Year           int64
position       int64
Team          object
matches        int64
wins           int64
draws          int64
loses          int64
scored         int64
pts            int64
xG           float64
xGA          float64
```

```
%LoseR        float64
%DrawR        float64
Shots         float64
Yellow        float64
Red           float64
Fouls         float64
S_OnTarget    float64
dtype: object
```

Similarly to our first dataset, the second dataset contains mostly numerical data representing totals in each listed statistical category accross an entire season. Also similarly to our first dataset, the exception to this rule is the league and team name features which are categorical variables and represented as strings.

```
print("The first dataset contains",df.size,"elements and",df.shape[0],"rows.")
```

```
The first dataset contains 506604 elements and 12062 rows.
```

```
print("The second dataset contains",df2.size,"elements and",df2.shape[0],"rows.")
```

```
The second dataset contains 9310 elements and 490 rows.
```

```
# check for null values
df.isnull().sum()
```

```
Unnamed: 0                     0
Home Team                      0
Away Team                      0
Score                          0
Half Time Score                0
Match Excitement               0
Home Team Rating               0
Away Team Rating               0
Home Team Possession %         0
Away Team Possession %         0
Home Team Off Target Shots     0
Home Team On Target Shots      0
Home Team Total Shots          0
```

```
Home Team Blocked Shots         0
Home Team Corners               0
Home Team Throw Ins             0
Home Team Pass Success %        0
Home Team Aerials Won           0
Home Team Clearances            0
Home Team Fouls                 0
Home Team Yellow Cards          0
Home Team Second Yellow Cards   0
Home Team Red Cards             0
Away Team Off Target Shots      0
Away Team On Target Shots       0
Away Team Total Shots           0
Away Team Blocked Shots         0
Away Team Corners               0
Away Team Throw Ins             0
Away Team Pass Success %        0
Away Team Aerials Won           0
Away Team Clearances            0
Away Team Fouls                 0
Away Team Yellow Cards          0
Away Team Second Yellow Cards   0
Away Team Red Cards             0
Home Team Goals Scored          0
Away Team Goals Scored          0
Home Team Goals Conceeded       0
Away Team Goals Conceeded       0
year                            0
league                          0
dtype: int64
```

```
# check for null values
df2.isnull().sum()
```

```
League       0
Year         0
position     0
Team         0
matches      0
wins         0
draws        0
```

```
loses         0
scored        0
pts           0
xG            0
xGA           0
%LoseR        6
%DrawR        6
Shots         6
Yellow        6
Red           6
Fouls         6
S_OnTarget    6
dtype: int64
```

```
# using this function, we can see that the "null" values are coming from the last several rows in the dataset
df_null = df2.isnull()
print (df_null)
```

```
    League   Year  position   Team  matches   wins  draws  loses  scored  \
0    False  False     False  False    False  False  False  False   False
1    False  False     False  False    False  False  False  False   False
2    False  False     False  False    False  False  False  False   False
3    False  False     False  False    False  False  False  False   False
4    False  False     False  False    False  False  False  False   False
..     ...    ...       ...    ...      ...    ...    ...    ...     ...
485  False  False     False  False    False  False  False  False   False
486  False  False     False  False    False  False  False  False   False
487  False  False     False  False    False  False  False  False   False
488  False  False     False  False    False  False  False  False   False
489  False  False     False  False    False  False  False  False   False

       pts     xG    xGA  %LoseR  %DrawR  Shots  Yellow    Red  Fouls  \
0    False  False  False   False   False  False   False  False  False
1    False  False  False   False   False  False   False  False  False
2    False  False  False   False   False  False   False  False  False
3    False  False  False   False   False  False   False  False  False
4    False  False  False   False   False  False   False  False  False
..     ...    ...    ...     ...     ...    ...     ...    ...    ...
485  False  False  False    True    True   True    True   True   True
486  False  False  False    True    True   True    True   True   True
487  False  False  False    True    True   True    True   True   True
```

```
488  False  False  False    True    True   True    True   True   True
489  False  False  False    True    True   True    True   True   True

     S_OnTarget
0         False
1         False
2         False
3         False
4         False
..          ...
485        True
486        True
487        True
488        True
489        True

[490 rows x 19 columns]
```

```
# this function shows the last rows where we are seeing missing data
df2.tail(6)
```

```
# imputes missing %LoseR values
position = 484

while position < 490:
  tempMatches = df2.at[position,"matches"]
  tempLoses = df2.at[position,"loses"]
  newNumber = tempLoses/tempMatches
  df2.at[position,"%LoseR"] = newNumber
  position+=1


# imputes missing %LoseR values
position = 484

while position < 490:
  tempMatches = df2.at[position,"matches"]
  tempDraws = df2.at[position,"draws"]
  newNumber = tempDraws/tempMatches
  df2.at[position,"%DrawR"] = newNumber
  position+=1


# shows the newly inputed values
df2.tail(6)
```

From the isnull() functions above, we can see that our first dataset does not contain any missing or null values; however, our second dataset does have some missing values specifically within the last 6 rows and on the following features: %LoseR, %DrawR, Shots, Yellow, Red, Fouls, S_OnTarget. In the code sections following the isnull() function, I was able to impute the remaining values in both the %LoseR and %DrawR columns using the data on matches, loses, and draws in each respective row. This will help enhance the completeness of our final dataset and make our final model more accurate. The remaining missing values were not able to be simply imputed because there was no data on fouls or missed shots within each row.

## 3. Data Description

- Create a data description (data dictionary) for your data sets.
    - Describe each variable
    - If categorical, what levels are present? If the levels are encoded, what do the codes mean?
    - If numeric, provide min, max, median and any other univariate stats you'd like to add in.
- Where appropriate, provide histograms or other visualizations to characterize each variable.

```
# Let's start with the first dataset
df.head(5)
```

```
# Lists the datatypes of the attributes within the first dataset
df.dtypes
```

```
Unnamed: 0                          int64
Home Team                          object
Away Team                          object
Score                              object
Half Time Score                    object
Match Excitement                  float64
Home Team Rating                  float64
Away Team Rating                  float64
Home Team Possession %              int64
Away Team Possession %              int64
Home Team Off Target Shots        float64
Home Team On Target Shots         float64
Home Team Total Shots             float64
Home Team Blocked Shots           float64
Home Team Corners                 float64
Home Team Throw Ins               float64
Home Team Pass Success %          float64
Home Team Aerials Won             float64
Home Team Clearances              float64
Home Team Fouls                   float64
Home Team Yellow Cards            float64
Home Team Second Yellow Cards     float64
Home Team Red Cards               float64
Away Team Off Target Shots        float64
Away Team On Target Shots         float64
```

```
Away Team Total Shots           float64
Away Team Blocked Shots         float64
Away Team Corners               float64
Away Team Throw Ins             float64
Away Team Pass Success %        float64
Away Team Aerials Won           float64
Away Team Clearances            float64
Away Team Fouls                 float64
Away Team Yellow Cards          float64
Away Team Second Yellow Cards   float64
Away Team Red Cards             float64
Home Team Goals Scored            int64
Away Team Goals Scored            int64
Home Team Goals Conceeded         int64
Away Team Goals Conceeded         int64
year                              int64
league                           object
dtype: object
```

Attribute Information:

- Unnamed: 0 : index
- Home Team: club name of team playing at home
- Away Team: club name of team playing on the road
- Score: final score of the game
- Match excitement: excitement rating of the match
  - Not entirely sure how they derived this. I am assuming a combination between attendance, crowd noise, and TV views
- Home team rating: match rating of the home team
- Away team rating: match rating of the away team
  - Team Rating note: I am assuming this is a calculated value of how well a team performed in a match
- Home team possession %: percent of the match the home team had possession of the ball
- Away team possession %: percent of the match the away team had possession of the ball
- Home Team Off Target Shots: number of shots off target for the home team
- Home Team On Target Shots: number of shots on target for the home team

- Home Team Total Shots: total number of shots for the home team
- Home Team Blocked Shots: number of blocked shots by the home team
- Home Team Corners: number of corners for the home team
- Home Team Throw Ins: number of throw ins for the home team
- Home Team Pass Success %: percent of successful passes for the home team
- Home Team Aerials Won: number of balls won in the air by the home team
- Home Team Clearances: number of balls cleared by the home team
- Home Team Fouls: number of fouls committed by the home team
- Home Team Yellow Cards: number of yellow cards for the home team
- Home Team Second Yellow Cards: number of times a second yellow card is given to a player
- Home Team Red Cards: number of red cards for the home team
- Away Team Off Target Shots: number of shots off target for the away team
- Away Team On Target Shots: number of shots on target for the away team
- Away Team Total Shots: total number of shots for the away team
- Away Team Blocked Shots: number of blocked shots by the away team
- Away Team Corners: number of corners for the away team
- Away Team Throw Ins: number of throw ins for the away team
- Away Team Pass Success %: percent of successful passes for the away team
- Away Team Aerials Won: number of balls won in the air by the away team
- Away Team Clearances: number of balls cleared by the away team
- Away Team Fouls: number of fouls committed by the away team
- Away Team Yellow Cards: number of yellow cards for the away team
- Away Team Second Yellow Cards: number of times a second yellow card is given to a player
- Away Team Red Cards: number of red cards for the away team
- Home Team Goals Scored: number of goals scored by the home team
- Away Team Goals Scored: number of goals scored by the away team
- Home Team Goals Conceeded: number of goals conceded by the home team
- Away Team Goals Conceeded: number of goals conceded by the away team

- Year: year
- League: soccer league

Categorical variables:

- Home team
- Away team
- Score
- League

```
# Let's look at the first dataset's numerical variables
df.describe()
```

```
# Now let's look at the second dataset
df2.head(5)
```

```
# Lists the datatypes of the attributes within the second dataset
df2.dtypes
```

```
League         object
Year            int64
position        int64
Team           object
matches         int64
wins            int64
draws           int64
loses           int64
scored          int64
pts             int64
xG            float64
xGA           float64
%LoseR        float64
%DrawR        float64
Shots         float64
Yellow        float64
```

```
Red            float64
Fouls          float64
S_OnTarget     float64
dtype: object
```

Attribute Information:

- League: league
- Year: year
- Position: finishing position in that league for that year
- Team: club name
- Matches: matches played
- Wins: wins
- Draws: draws/ties
- Loses: loses
- Scored: goals for
- Pts: points allocated for wins (+3) and draws (+1)
- xG: expected goals for
- xGA: expected goals against
- %LoseR: % games lost
- %DrawR: % games drawn
- Shots: shots
- Yellow: yellow cards
- Red: red cards
- Fouls: fouls committed
- S_OnTarget: shots on target

```
# Let's look at the second dataset's numerical variables
df2.describe()
```

## 4. Merge Datasets

Now that you have a better feel for each of your two (or three, for the 7394 students) data sets, it is time to merge them. Describe your strategy for merging the data sets and then actually perform the merge.

Develop a strategy for verifying that the data is properly merged (hoping and finger-crossing are not valid strategies).

```
df.head(2)
```

```
df2.head(2)
```

For our model, we are planning to build a model that predicts the number of goals scored for both the home and away team. To accomplish this, we are going to build two seperate models:

1) Predict home team score

2) Predict away team score

Following this process, our idea for merging is:

1) Copy 'df' into a home and away dataset

2) Copy 'df2' into two other datasets, called 'df2_<home/away>'

3) Rename the Team column to home or away team, respectively, of 'df2_<home/away>'

4) Change the values of Home/Away Team column to match with the syntax of the first dataset

example) Manchester United --> MAN UTD

```
teams = df['Home Team'].unique()
```

```
teams_df2 = df2['Team'].unique()
```

```
teams = np.sort(teams)
```

```
teams_df2 = np.sort(teams_df2)
```

```
teamsAcrossDatasets = {}
for t in teams_df2:
    temp = t
    t = t.upper()
    if t in teams:
        teamsAcrossDatasets[temp] = t
    else:
        teamsAcrossDatasets[temp] = "Need to find"
```

```
teamsAcrossDatasets
```

```
{'AC Milan': 'Need to find',
 'Alaves': 'Need to find',
 'Almeria': 'Need to find',
 'Amiens': 'AMIENS',
 'Angers': 'ANGERS',
 'Arsenal': 'ARSENAL',
 'Aston Villa': 'ASTON VILLA',
 'Atalanta': 'ATALANTA',
 'Athletic Club': 'Need to find',
 'Atletico Madrid': 'ATLETICO MADRID',
 'Augsburg': 'AUGSBURG',
 'Barcelona': 'BARCELONA',
 'Bayer Leverkusen': 'Need to find',
 'Bayern Munich': 'Need to find',
 'Benevento': 'BENEVENTO',
 'Bologna': 'BOLOGNA',
```

```
    'Bordeaux': 'BORDEAUX',
    'Borussia Dortmund': 'Need to find',
    'Borussia M.Gladbach': 'Need to find',
    'Bournemouth': 'BOURNEMOUTH',
    'Brighton': 'BRIGHTON',
    'Burnley': 'BURNLEY',
    'Caen': 'CAEN',
    'Cagliari': 'CAGLIARI',
    'Cardiff': 'CARDIFF',
    'Carpi': 'CARPI',
    'Celta Vigo': 'Need to find',
    'Cesena': 'CESENA',
    'Chelsea': 'CHELSEA',
    'Chievo': 'CHIEVO',
    'Cordoba': 'Need to find',
    'Crotone': 'CROTONE',
    'Crystal Palace': 'CRYSTAL PALACE',
    'Darmstadt': 'DARMSTADT',
    'Deportivo La Coruna': 'Need to find',
    'Dijon': 'DIJON',
    'Eibar': 'EIBAR',
    'Eintracht Frankfurt': 'Need to find',
    'Elche': 'ELCHE',
    'Empoli': 'EMPOLI',
    'Espanyol': 'ESPANYOL',
    'Everton': 'EVERTON',
    'Evian Thonon Gaillard': 'Need to find',
    'FC Cologne': 'Need to find',
    'Fiorentina': 'FIORENTINA',
    'Fortuna Duesseldorf': 'Need to find',
    'Freiburg': 'FREIBURG',
    'Frosinone': 'FROSINONE',
    'Fulham': 'FULHAM',
    'GFC Ajaccio': 'GFC AJACCIO',
    'Genoa': 'GENOA',
    'Getafe': 'GETAFE',
    'Girona': 'GIRONA',
    'Granada': 'GRANADA',
    'Guingamp': 'GUINGAMP',
    'Hamburger SV': 'Need to find',
    'Hannover 96': 'Need to find',
    'Hertha Berlin': 'Need to find'
```

```
teamsNeeded = [k for k,v in teamsAcrossDatasets.items() if v == 'Need to find']


# Imputed off of prior football knowledge and sources listed below
# For Evian Thonon Gaillard and THONON ÉVIAN: https://en.wikipedia.org/wiki/Thonon_Evian_Grand_Gen%C3%A8ve_F.C.
# For HSV and Hamburger SV: https://www.hsv.de/en/homepage
# For Lens and RC LENS: https://en.wikipedia.org/wiki/RC_Lens
# For Reims and STADE DE REIMS: https://en.wikipedia.org/wiki/Stade_de_Reims

otherTeams = ['MILAN','ALAVÉS', 'ALMERÍA', 'ATHLETIC', 'LEVERKUSEN','BAYERN','DORTMUND',"M'GLADBACH",
              'CELTA', 'CÓRDOBA','DEPORTIVO','FRANKFURT','THONON ÉVIAN', '1. FC KÖLN', 'DÜSSELDORF',
              'HSV', 'HANNOVER', 'HERTHA', 'HULL CITY', 'LEGANÉS', 'LEICESTER CITY', 'RC LENS','MAINZ',
             'MÁLAGA', 'MAN CITY','MAN UTD','FC METZ','NEWCASTLE','NÜRNBERG','PSG','PARMA','QPR','RB LEIPZIG',
             'VALLADOLID', 'STADE DE REIMS', 'STADE RENNAIS', 'HUESCA', 'SPAL','SAINT-ÉTIENNE','SCHALKE','SEVILLA FC',
             'GIJÓN','HELLAS','STUTTGART','W. BREMEN','WEST BROM','WOLVES']


for i, t in enumerate(teamsNeeded):
    teamsAcrossDatasets[t] = otherTeams[i]


teamsAcrossDatasets
```

```
{'AC Milan': 'MILAN',
 'Alaves': 'ALAVÉS',
 'Almeria': 'ALMERÍA',
 'Amiens': 'AMIENS',
 'Angers': 'ANGERS',
 'Arsenal': 'ARSENAL',
 'Aston Villa': 'ASTON VILLA',
 'Atalanta': 'ATALANTA',
 'Athletic Club': 'ATHLETIC',
 'Atletico Madrid': 'ATLETICO MADRID',
 'Augsburg': 'AUGSBURG',
 'Barcelona': 'BARCELONA',
 'Bayer Leverkusen': 'LEVERKUSEN',
 'Bayern Munich': 'BAYERN',
 'Benevento': 'BENEVENTO',
 'Bologna': 'BOLOGNA',
 'Bordeaux': 'BORDEAUX',
 'Borussia Dortmund': 'DORTMUND',
```

```
        'Borussia M.Gladbach': "M'GLADBACH",
        'Bournemouth': 'BOURNEMOUTH',
        'Brighton': 'BRIGHTON',
        'Burnley': 'BURNLEY',
        'Caen': 'CAEN',
        'Cagliari': 'CAGLIARI',
        'Cardiff': 'CARDIFF',
        'Carpi': 'CARPI',
        'Celta Vigo': 'CELTA',
        'Cesena': 'CESENA',
        'Chelsea': 'CHELSEA',
        'Chievo': 'CHIEVO',
        'Cordoba': 'CÓRDOBA',
        'Crotone': 'CROTONE',
        'Crystal Palace': 'CRYSTAL PALACE',
        'Darmstadt': 'DARMSTADT',
        'Deportivo La Coruna': 'DEPORTIVO',
        'Dijon': 'DIJON',
        'Eibar': 'EIBAR',
        'Eintracht Frankfurt': 'FRANKFURT',
        'Elche': 'ELCHE',
        'Empoli': 'EMPOLI',
        'Espanyol': 'ESPANYOL',
        'Everton': 'EVERTON',
        'Evian Thonon Gaillard': 'THONON ÉVIAN',
        'FC Cologne': '1. FC KÖLN',
        'Fiorentina': 'FIORENTINA',
        'Fortuna Duesseldorf': 'DÜSSELDORF',
        'Freiburg': 'FREIBURG',
        'Frosinone': 'FROSINONE',
        'Fulham': 'FULHAM',
        'GFC Ajaccio': 'GFC AJACCIO',
        'Genoa': 'GENOA',
        'Getafe': 'GETAFE',
        'Girona': 'GIRONA',
        'Granada': 'GRANADA',
        'Guingamp': 'GUINGAMP',
        'Hamburger SV': 'HSV',
        'Hannover 96': 'HANNOVER',
        'Hertha Berlin': 'HERTHA',
```

```
df.rename(columns = {'year':'Year'}, inplace = True)
```

```
df_home = df.copy()
df_away = df.copy()


def convert_to_common_team_name(team):
    return teamsAcrossDatasets[team]

def transformDF2(df2):
    df2_home = df2.copy()
    df2_away = df2.copy()
    df2_home['Team'] = df2_home['Team'].apply(convert_to_common_team_name)
    df2_away['Team'] = df2_away['Team'].apply(convert_to_common_team_name)
    df2_home.rename(columns = {'Team':'Home Team'}, inplace = True)
    df2_away.rename(columns = {'Team':'Away Team'}, inplace = True)
    return df2_home, df2_away


df2_home, df2_away = transformDF2(df2)


aggData = pd.merge(df2_home, df_home, on=['Year', 'Home Team'], how='inner')
aggData.head(5)
```

```
len(aggData)
```

```
8516
```

```
aggData.iloc[8510]
```

```
League                                  Serie_A
Year                                       2018
position                                     16
Home Team                                 PARMA
matches                                      38
wins                                         10
draws                                        11
loses                                        17
scored                                       41
pts                                          41
xG                                    41.098644
xGA                                   64.981144
%LoseR                                 0.447368
%DrawR                                 0.289474
Shots                                       NaN
Yellow                                      NaN
Red                                         NaN
Fouls                                       NaN
S_OnTarget                                  NaN
Unnamed: 0                                 8523
Away Team                                 GENOA
Score                                       1-0
Half Time Score                             0-0
Match Excitement                            3.6
Home Team Rating                            6.8
Away Team Rating                            5.8
Home Team Possession %                       41
Away Team Possession %                       59
Home Team Off Target Shots                  5.0
Home Team On Target Shots                   3.0
Home Team Total Shots                       9.0
```

```
Home Team Blocked Shots                1.0
Home Team Corners                      4.0
Home Team Throw Ins                   14.0
Home Team Pass Success %              76.0
Home Team Aerials Won                 16.0
Home Team Clearances                  24.0
Home Team Fouls                       12.0
Home Team Yellow Cards                 2.0
Home Team Second Yellow Cards          0.0
Home Team Red Cards                    0.0
Away Team Off Target Shots             9.0
Away Team On Target Shots              1.0
Away Team Total Shots                 13.0
Away Team Blocked Shots                3.0
Away Team Corners                      5.0
Away Team Throw Ins                   32.0
Away Team Pass Success %              83.0
Away Team Aerials Won                 17.0
Away Team Clearances                   8.0
Away Team Fouls                       12.0
Away Team Yellow Cards                 1.0
Away Team Second Yellow Cards          0.0
Away Team Red Cards                    0.0
Home Team Goals Scored                   1
Away Team Goals Scored                   0
Home Team Goals Conceeded                0
Away Team Goals Conceeded                1
```

```
aggData.iloc[1]
```

```
League                                  La_liga
Year                                       2014
position                                      1
Home Team                             BARCELONA
matches                                      38
wins                                         30
draws                                         4
loses                                         4
scored                                      110
pts                                          94
xG                                   102.980152
xGA                                   28.444293
```

```
%LoseR                                0.25
%DrawR                            0.714286
Shots                                626.0
Yellow                                66.0
Red                                    3.0
Fouls                                369.0
S_OnTarget                           273.0
Unnamed: 0                            9423
Away Team                         ATHLETIC
Score                                  2-0
Half Time Score                        0-0
Match Excitement                       4.3
Home Team Rating                       7.7
Away Team Rating                       5.7
Home Team Possession %                  61
Away Team Possession %                  39
Home Team Off Target Shots             5.0
Home Team On Target Shots              8.0
Home Team Total Shots                 14.0
Home Team Blocked Shots                1.0
Home Team Corners                      5.0
Home Team Throw Ins                   20.0
Home Team Pass Success %              84.0
Home Team Aerials Won                 12.0
Home Team Clearances                  11.0
Home Team Fouls                       12.0
Home Team Yellow Cards                 1.0
Home Team Second Yellow Cards          0.0
Home Team Red Cards                    0.0
Away Team Off Target Shots             1.0
Away Team On Target Shots              2.0
Away Team Total Shots                  3.0
Away Team Blocked Shots                0.0
Away Team Corners                      1.0
Away Team Throw Ins                   22.0
Away Team Pass Success %              74.0
Away Team Aerials Won                 15.0
Away Team Clearances                  19.0
Away Team Fouls                       11.0
Away Team Yellow Cards                 1.0
Away Team Second Yellow Cards          0.0
Away Team Red Cards                    0.0
```

```
Home Team Goals Scored                2
Away Team Goals Scored                0
Home Team Goals Conceeded             0
```

The merge looks good!

From looking at this, columns that could be droped for a model:

df: Goals conceded, Total Shots, Unnamed, league,

## 5. Explore Bivariate Relationships

- Choose a reasoned set of variables to explore further. You don't have to explore all possible pairs of variables, nor do we want to grade that much. Choose 7 - 9 variables. One should be a variable that you'd like to predict (target variable) using the others (predictor variables).
- List your predictor variables
- List your target variable
- Briefly describe why you have chosen these.

Use any of the available visualizations from Seaborn to explore the relationships between the variables. Explore the relationships among the predictor variables as well as the relationship between each predictor variable and the target variable. Which of the predictor variables are most strongly related? Are there any interesting relationships between categorical predictors and numeric predictors? If there are any dichotomous variables, does that influence any of the relationships? Are the relationships positive or negative?

Below each plot, you should provide a description and interpretation of the plot. Make sure to include why the variables in that plot were chosen and what you hope the reader would gain from it as well.

```
# correlation heat map giving correlation between single game statistics and overall match excitement
import seaborn as sn
```

```
corrMatt = aggData[["Match Excitement", "Home Team Clearances",
                    "Home Team Fouls", "Home Team Total Shots",
                    "Home Team Corners", "Home Team Throw Ins",
                    "Home Team Rating", "Home Team Aerials Won"]].corr()
mask = np.array(corrMatt)
mask[np.tril_indices_from(mask)] = False
sn.heatmap(corrMatt, mask=mask,
           vmax=.8, square=True,annot=True)
```

The above correlation heatmap finds the correlation between numerous different individual match statistics and the overall excitement rating. In creating this correlation map, we used all of the home team statistics because most of the fans at any given game will likely be favoring the home team and we believed that the home team statistics would be an overall better predictor of match excitement. Unsurprisingly, the statistics with the biggest positive correlation with match excitement was total shots with both clearances and total throw-ins exhibiting a slight negative correlation with match excitement. Another correlation that logically makes sense was that total corners was heavily correlated with total shots because most corners are generated through shots that the goalie blocks out of bounds.

```
# correlation heat map giving correlation between season statistics and total wins

corrMatt = aggData[["wins","Shots",
                    "Yellow","Red",
                    "Fouls","S_OnTarget",
                    "matches", "scored"]].corr()
mask = np.array(corrMatt)
mask[np.tril_indices_from(mask)] = False
sn.heatmap(corrMatt, mask=mask,
           vmax=.8, square=True,annot=True)
```

Based on the heatmap above, the number of goals scored by a team throughout the season seems to correlate most closely with the total number of wins by that team followed closely by the total number of shots taken. This makes sense as intuitively I would assume that goals and shots would create more of a difference in win percentage than other things such as yellow cards, red cards, or total fouls. I did find it interesting; however, that yellow cards, red cards, and total fouls did have a decently sized negative correlation with total wins indicating that these statistics did negatively impact total wins despite fouls being a critical and sometimes intentional part of the game.

```
# Get bottom teams
bottomTeams = aggData.where(aggData['position'] > 15)
bottomTeams = bottomTeams.dropna()


# Get top teams
topTeams = aggData.where(aggData['position'] <= 5)
topTeams = topTeams.dropna()


# displays avg of different season totals for teams in the top 15 vs bottom 5

print("Bottom 5 Team Averages:")
print("   Shots:",sum(bottomTeams["Shots"])/len(bottomTeams["Shots"]))
print("   Yellow Cards:",sum(bottomTeams["Yellow"])/len(bottomTeams["Yellow"]))
print("   Red Cards:",sum(bottomTeams["Red"])/len(bottomTeams["Red"]))
print("   Total Fouls:",sum(bottomTeams["Fouls"])/len(bottomTeams["Fouls"]))
print("   Shots on target:",sum(bottomTeams["S_OnTarget"])/len(bottomTeams["S_OnTarget"]))
print("   Total goals scored:",sum(bottomTeams["scored"])/len(bottomTeams["scored"]))

print("Top 5 Team Averages:")
print("   Shots:",sum(topTeams["Shots"])/len(topTeams["Shots"]))
print("   Yellow Cards:",sum(topTeams["Yellow"])/len(topTeams["Yellow"]))
print("   Red Cards:",sum(topTeams["Red"])/len(topTeams["Red"]))
print("   Total Fouls:",sum(topTeams["Fouls"])/len(topTeams["Fouls"]))
print("   Shots on target:",sum(topTeams["S_OnTarget"])/len(topTeams["S_OnTarget"]))
print("   Total goals scored:",sum(topTeams["scored"])/len(topTeams["scored"]))
```

```
Bottom 5 Team Averages:
   Shots: 410.5486111111111
   Yellow Cards: 83.01636904761905
   Red Cards: 4.838293650793651
   Total Fouls: 510.35515873015873
   Shots on target: 133.7470238095238
```

```
        Total goals scored: 35.7281746031746
     Top 5 Team Averages:
        Shots: 544.600464037123
        Yellow Cards: 70.32621809744779
        Red Cards: 3.266357308584687
        Total Fouls: 454.6324825986079
        Shots on target: 207.51647331786543
        Total goals scored: 73.25197215777263
```

The output of the code above shows the statistical averages of teams that are placed within the top 5 teams in a league in any given year versus the teams that are in the bottom 5. The results of this experiment are not very surprising with the teams at the top experiencing more shots and goals scored while getting called for less fouls than the teams in the bottom of the league.

```
# bar chart comparing average match excitment accross the 5 major leagues

sn.barplot(data=aggData,x='League',y='Match Excitement')
```

The output of the code above is a barchart representing the different average match excitment levels amongst the top 5 soccer leagues in the world. The line at the top of each bar serves to present the level of uncertainty around each estimation. As you can see, Ligue 1 (the French soccer league) appears to have the lowest average match excitement amongst the top 5 leagues. This isn't surprising as the French league is generally considered to be the least competitive and generally worst league amongst the top 5. However, what is surprising is that the English Premier League (EPL) which is considered to be the most competive league in the top 5 has a lower average excitement than both the Bundesliga (German soccer league) and Serie A (Italian soccer league) which are both generally considered to be less competitive. Overall, all of the averages are very similar meaning we can't take too much from these results; however, I do still think that comparing the leagues in this way is interesting. Additionally, I would like to know more about how the match excitement statistic was created and hopefully this can give us more insight into why we are seeing the differences amongst each league that are displayed above.

Ultimate model plan:

- Predict scores for 2019 and 2020 (target variables)
- Will attempt to use both past season statistics as well as past single-game statistics as our predictor variables
- Two models: one for home goals and one for away
- Eval on accuracy of score, goal differential, and match winner

## 6. References

[1] Football Data : Top 5 Leagues. https://www.kaggle.com/sanjeetsinghnaik/football-data-top-5-leagues

[2] Points Table: Top 5 Leagues. https://www.kaggle.com/thegreatcoder/points-table-of-5-leagues-in-football-20142018

1s completed at 11:17 AM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.