```
# Format:
# 1) Imports
# 2) Load CSVs
# 3) Data Structures
# 4) Data Preprocessing Helper Functions
# 5) Create Training Set

############################## IMPORTS ##############################

from __future__ import division
import pandas as pd
import numpy as np
import os.path
import math
import collections
import sklearn

############################## LOAD CSVs ##############################

reg_season_compact_pd = pd.read_csv('Data/KaggleData/RegularSeasonCompactResults.csv')
teams_pd = pd.read_csv('Data/KaggleData/Teams.csv')
tourney_compact_pd = pd.read_csv('Data/KaggleData/NCAATourneyCompactResults.csv')
conference_pd = pd.read_csv('Data/KaggleData/Conference.csv')
tourney_results_pd = pd.read_csv('Data/KaggleData/TourneyResults.csv')
tourney_seeds_pd = pd.read_csv('Data/KaggleData/NCAATourneySeeds.csv')
team_conferences_pd = pd.read_csv('Data/KaggleData/TeamConferences.csv')

############################## DATA STRUCTURES ##############################

teamList = teams_pd['TeamName'].tolist()
NCAAChampionsList = tourney_results_pd['NCAA Champion'].tolist()

############################## HELPER FUNCTIONS ##############################

def checkPower6Conference(team_id):
    team_pd = team_conferences_pd[(team_conferences_pd['Season'] == 2018) &
(team_conferences_pd['TeamID'] == team_id)]
    # Can't find the team
    if (len(team_pd) == 0):
        return 0
    confName = team_pd.iloc[0]['ConfAbbrev']
    return int(confName == 'sec' or confName == 'acc'or confName == 'big_ten' or confName ==
'big_twelve' or confName == 'big_east' or confName == 'pac_twelve')

def getTeamID(name):
    return teams_pd[teams_pd['TeamName'] == name].values[0][0]

def getTeamName(team_id):
    return teams_pd[teams_pd['TeamID'] == team_id].values[0][1]

def getNumChampionships(team_id):
    name = getTeamName(team_id)
    return NCAAChampionsList.count(name)

def getListForURL(team_list):
    team_list = [x.lower() for x in team_list]
    team_list = [t.replace(' ', '-') for t in team_list]
    team_list = [t.replace('st', 'state') for t in team_list]
    team_list = [t.replace('northern-dakota', 'north-dakota') for t in team_list]
    team_list = [t.replace('nc-', 'north-carolina-') for t in team_list]
    team_list = [t.replace('fl-', 'florida-') for t in team_list]
    team_list = [t.replace('ga-', 'georgia-') for t in team_list]
    team_list = [t.replace('lsu', 'louisiana-state') for t in team_list]
    team_list = [t.replace('maristate', 'marist') for t in team_list]
    team_list = [t.replace('stateate', 'state') for t in team_list]
    team_list = [t.replace('northernorthern', 'northern') for t in team_list]
    team_list = [t.replace('usc', 'southern-california') for t in team_list]
    base = 'http://www.sports-reference.com/cbb/schools/'
    for team in team_list:
        url = base + team + '/'
getListForURL(teamList);

def handleCases(arr):
    indices = []
    listLen = len(arr)
    for i in range(listLen):
        if (arr[i] == 'St' or arr[i] == 'FL'):
            indices.append(i)
    for p in indices:
        arr[p-1] = arr[p-1] + ' ' + arr[p]
    for i in range(len(indices)):
        arr.remove(arr[indices[i] - i])
    return arr

def checkConferenceChamp(team_id, year):
    year_conf_pd = conference_pd[conference_pd['Year'] == year]
    champs = year_conf_pd['Regular Season Champ'].tolist()
    # For handling cases where there is more than one champion
    champs_separated = [words for segments in champs for words in segments.split()]
    name = getTeamName(team_id)
    champs_separated = handleCases(champs_separated)
    if (name in champs_separated):
        return 1
    else:
        return 0

def checkConferenceTourneyChamp(team_id, year):
    year_conf_pd = conference_pd[conference_pd['Year'] == year]
    champs = year_conf_pd['Tournament Champ'].tolist()
    name = getTeamName(team_id)
    if (name in champs):
        return 1
    else:
        return 0

def getTourneyAppearances(team_id):
    return len(tourney_seeds_pd[tourney_seeds_pd['TeamID'] == team_id].index)

def handleDifferentCSV(df):
    # The stats CSV is a lit different in terms of naming so below is just some data cleaning
```

```
df['School'] = df['School'].replace('(State)', 'St', regex=True)
df['School'] = df['School'].replace('Albany (NY)', 'Albany NY')
df['School'] = df['School'].replace('Boston University', 'Boston Univ')
df['School'] = df['School'].replace('Central Michigan', 'C Michigan')
df['School'] = df['School'].replace('(Eastern)', 'E', regex=True)
df['School'] = df['School'].replace('Louisiana St', 'LSU')
df['School'] = df['School'].replace('North Carolina St', 'NC State')
df['School'] = df['School'].replace('Southern California', 'USC')
df['School'] = df['School'].replace('University of California', 'California', regex=True)
df['School'] = df['School'].replace('American', 'American Univ')
df['School'] = df['School'].replace('Arkansas-Little Rock', 'Ark Little Rock')
df['School'] = df['School'].replace('Arkansas-Pine Bluff', 'Ark Pine Bluff')
df['School'] = df['School'].replace('Bowling Green St', 'Bowling Green')
df['School'] = df['School'].replace('Brigham Young', 'BYU')
df['School'] = df['School'].replace('Cal Poly', 'Cal Poly SLO')
df['School'] = df['School'].replace('Centenary (LA)', 'Centenary')
df['School'] = df['School'].replace('Central Connecticut St', 'Central Conn')
df['School'] = df['School'].replace('Charleston Southern', 'Charleston So')
df['School'] = df['School'].replace('Coastal Carolina', 'Coastal Car')
df['School'] = df['School'].replace('College of Charleston', 'Col Charleston')
df['School'] = df['School'].replace('Cal St Fullerton', 'CS Fullerton')
df['School'] = df['School'].replace('Cal St Sacramento', 'CS Sacramento')
df['School'] = df['School'].replace('Cal St Bakersfield', 'CS Bakersfield')
df['School'] = df['School'].replace('Cal St Northridge', 'CS Northridge')
df['School'] = df['School'].replace('East Tennessee St', 'ETSU')
df['School'] = df['School'].replace('Detroit Mercy', 'Detroit')
df['School'] = df['School'].replace('Fairleigh Dickinson', 'F Dickinson')
df['School'] = df['School'].replace('Florida Atlantic', 'FL Atlantic')
df['School'] = df['School'].replace('Florida Gulf Coast', 'FL Gulf Coast')
df['School'] = df['School'].replace('Florida International', 'Florida Intl')
df['School'] = df['School'].replace('George Washington', 'G Washington')
df['School'] = df['School'].replace('Georgia Southern', 'Ga Southern')
df['School'] = df['School'].replace('Gardner-Webb', 'Gardner Webb')
df['School'] = df['School'].replace('Illinois-Chicago', 'IL Chicago')
df['School'] = df['School'].replace('Kent St', 'Kent')
df['School'] = df['School'].replace('Long Island University', 'Long Island')
df['School'] = df['School'].replace('Loyola Marymount', 'Loy Marymount')
df['School'] = df['School'].replace('Loyola (MD)', 'Loyola MD')
df['School'] = df['School'].replace('Loyola (IL)', 'Loyola-Chicago')
df['School'] = df['School'].replace('Massachusetts', 'MA Lowell')
df['School'] = df['School'].replace('Maryland-Eastern Shore', 'MD E Shore')
df['School'] = df['School'].replace('Miami (FL)', 'Miami FL')
df['School'] = df['School'].replace('Miami (OH)', 'Miami OH')
df['School'] = df['School'].replace('Missouri-Kansas City', 'Missouri KC')
df['School'] = df['School'].replace('Monmouth', 'Monmouth NJ')
df['School'] = df['School'].replace('Mississippi Valley St', 'MS Valley St')
df['School'] = df['School'].replace('Montana St', 'MTSU')
df['School'] = df['School'].replace('Northern Colorado', 'N Colorado')
df['School'] = df['School'].replace('North Dakota St', 'N Dakota St')
df['School'] = df['School'].replace('Northern Illinois', 'N Illinois')
df['School'] = df['School'].replace('Northern Kentucky', 'N Kentucky')
df['School'] = df['School'].replace('North Carolina A&T', 'NC A&T')
df['School'] = df['School'].replace('North Carolina Central', 'NC Central')
df['School'] = df['School'].replace('Pennsylvania', 'Penn')
df['School'] = df['School'].replace('South Carolina St', 'S Carolina St')
```

```
    df['School'] = df['School'].replace('Southern Illinois', 'S Illinois')
    df['School'] = df['School'].replace('UC-Santa Barbara', 'Santa Barbara')
    df['School'] = df['School'].replace('Southeastern Louisiana', 'SE Louisiana')
    df['School'] = df['School'].replace('Southeast Missouri St', 'SE Missouri St')
    df['School'] = df['School'].replace('Stephen F. Austin', 'SF Austin')
    df['School'] = df['School'].replace('Southern Methodist', 'SMU')
    df['School'] = df['School'].replace('Southern Mississippi', 'Southern Miss')
    df['School'] = df['School'].replace('Southern', 'Southern Univ')
    df['School'] = df['School'].replace('St. Bonaventure', 'St Bonaventure')
    df['School'] = df['School'].replace('St. Francis (NY)', 'St Francis NY')
    df['School'] = df['School'].replace('Saint Francis (PA)', 'St Francis PA')
    df['School'] = df['School'].replace('St. John\'s (NY)', 'St John\'s')
    df['School'] = df['School'].replace('Saint Joseph\'s', 'St Joseph\'s PA')
    df['School'] = df['School'].replace('Saint Louis', 'St Louis')
    df['School'] = df['School'].replace('Saint Mary\'s (CA)', 'St Mary\'s CA')
    df['School'] = df['School'].replace('Mount Saint Mary\'s', 'Mt St Mary\'s')
    df['School'] = df['School'].replace('Saint Peter\'s', 'St Peter\'s')
    df['School'] = df['School'].replace('Texas A&M-Corpus Christian', 'TAM C. Christian')
    df['School'] = df['School'].replace('Texas Christian', 'TCU')
    df['School'] = df['School'].replace('Tennessee-Martin', 'TN Martin')
    df['School'] = df['School'].replace('Texas-Rio Grande Valley', 'UTRGV')
    df['School'] = df['School'].replace('Texas Southern', 'TX Southern')
    df['School'] = df['School'].replace('Alabama-Birmingham', 'UAB')
    df['School'] = df['School'].replace('UC-Davis', 'UC Davis')
    df['School'] = df['School'].replace('UC-Irvine', 'UC Irvine')
    df['School'] = df['School'].replace('UC-Riverside', 'UC Riverside')
    df['School'] = df['School'].replace('Central Florida', 'UCF')
    df['School'] = df['School'].replace('Louisiana-Lafayette', 'ULL')
    df['School'] = df['School'].replace('Louisiana-Monroe', 'ULM')
    df['School'] = df['School'].replace('Maryland-Baltimore County', 'UMBC')
    df['School'] = df['School'].replace('North Carolina-Asheville', 'UNC Asheville')
    df['School'] = df['School'].replace('North Carolina-Greensboro', 'UNC Greensboro')
    df['School'] = df['School'].replace('North Carolina-Wilmington', 'UNC Wilmington')
    df['School'] = df['School'].replace('Nevada-Las Vegas', 'UNLV')
    df['School'] = df['School'].replace('Texas-Arlington', 'UT Arlington')
    df['School'] = df['School'].replace('Texas-San Antonio', 'UT San Antonio')
    df['School'] = df['School'].replace('Texas-El Paso', 'UTEP')
    df['School'] = df['School'].replace('Virginia Commonwealth', 'VA Commonwealth')
    df['School'] = df['School'].replace('Western Carolina', 'W Carolina')
    df['School'] = df['School'].replace('Western Illinois', 'W Illinois')
    df['School'] = df['School'].replace('Western Kentucky', 'WKU')
    df['School'] = df['School'].replace('Western Michigan', 'W Michigan')
    df['School'] = df['School'].replace('Abilene Christian', 'Abilene Chr')
    df['School'] = df['School'].replace('Montana State', 'Montana St')
    df['School'] = df['School'].replace('Central Arkansas', 'Cent Arkansas')
    df['School'] = df['School'].replace('Houston Baptist', 'Houston Bap')
    df['School'] = df['School'].replace('South Dakota St', 'S Dakota St')
    df['School'] = df['School'].replace('Maryland-Eastern Shore', 'MD E Shore')
    return df

def getHomeStat(row):
    if (row == 'H'):
        home = 1
    if (row == 'A'):
        home = -1
    if (row == 'N'):
        home = 0
    return home

def compareTwoTeams(id_1, id_2, year):
    team_1 = getSeasonData(id_1, year)
    team_2 = getSeasonData(id_2, year)
    diff = [a - b for a, b in zip(team_1, team_2)]
    return diff

def normalizeInput(arr):
    for i in range(arr.shape[1]):
        minVal = min(arr[:,i])
        maxVal = max(arr[:,i])
        arr[:,i] =  (arr[:,i] - minVal) / (maxVal - minVal)
    return arr

def normalizeInput2(X):
    return (X - np.mean(X, axis = 0)) / np.std(X, axis = 0)

############################## MAIN PREPROCESSING FUNCTIONS
##############################

def getSeasonData(team_id, year):
    stats_SOS_pd = pd.read_csv('Data/RegSeasonStats/MMStats_'+str(year)+'.csv')
    stats_SOS_pd = handleDifferentCSV(stats_SOS_pd)
    ratings_pd = pd.read_csv('Data/RatingStats/RatingStats_'+str(year)+'.csv')
    ratings_pd = handleDifferentCSV(ratings_pd)
    year_data_pd = reg_season_compact_pd[reg_season_compact_pd['Season'] == year]

    numFeatures = 16
    name = getTeamName(team_id)
    team = stats_SOS_pd[stats_SOS_pd['School'] == name]
    team_rating = ratings_pd[ratings_pd['School'] == name]
    if (len(team.index) == 0 or len(team_rating.index) == 0): #Can't find the team
        return [0 for x in range(numFeatures)]

    gamesWon = team['W'].values[0]
    gamesLost = team['L'].values[0]
    total3sMade = team['X3P'].values[0]
    totalTurnovers = 0 if math.isnan(team['TOV'].values[0]) else team['TOV'].values[0]
    totalAssists = 0 if math.isnan(team['AST'].values[0]) else team['AST'].values[0]
    totalRebounds = 0 if math.isnan(team['TRB'].values[0]) else team['TRB'].values[0]
    totalSteals = 0 if math.isnan(team['STL'].values[0]) else team['STL'].values[0]
    sos = team['SOS'].values[0]
    srs = team['SRS'].values[0]
    numWins = team['W'].values[0]
    totalPointsScored = team['Tm.'].values[0]

    totalPointsAllowed = team['Opp.'].values[0]
    # MM_Stats 1993-1995 don't have these stats so we need to get it from somewhere else
    if math.isnan(totalPointsAllowed):
        gamesPlayed = year_data_pd[(year_data_pd.WTeamID == team_id) | (year_data_pd.LTeamID ==
team_id)]
        totalPointsAllowed = gamesPlayed['LScore'].sum()
    #Finding tournament seed for that year
    tourneyYear = tourney_seeds_pd[tourney_seeds_pd['Season'] == year]
    seed = tourneyYear[tourneyYear['TeamID'] == team_id]
    if (len(seed.index) != 0):
        seed = seed.values[0][1]
        tournamentSeed = int(seed[1:3])
    else:
        tournamentSeed = 25 #Not sure how to represent if a team didn't make the tourney

    numGames = team['G'].values[0]

    avgPointsScored = totalPointsScored/numGames
    avgPointsAllowed = totalPointsAllowed/numGames
    avg3sMade = total3sMade/numGames
    avgTurnovers = totalTurnovers/numGames
    avgAssists = totalAssists/numGames
    avgRebounds = totalRebounds/numGames
    avgSteals = totalSteals/numGames
    return [numWins, avgPointsScored, avgPointsAllowed, checkPower6Conference(team_id),
avg3sMade, avgAssists, avgTurnovers,
           checkConferenceChamp(team_id, year), checkConferenceTourneyChamp(team_id, year),
tournamentSeed,
            sos, srs, avgRebounds, avgSteals, getTourneyAppearances(team_id),
getNumChampionships(team_id)]

def createSeasonDict(year):
    seasonDictionary = collections.defaultdict(list)
    for team in teamList:
        team_id = teams_pd[teams_pd['TeamName'] == team].values[0][0]
        team_vector = getSeasonData(team_id, year)
        seasonDictionary[team_id] = team_vector
    return seasonDictionary

def createTrainingSet(years, saveYears):
    totalNumGames = 0
    for year in years:
        season = reg_season_compact_pd[reg_season_compact_pd['Season'] == year]
        totalNumGames += len(season.index)
        tourney = tourney_compact_pd[tourney_compact_pd['Season'] == year]
        totalNumGames += len(tourney.index)
    numFeatures = len(getSeasonData(1181,2012)) #Just choosing a random team and seeing the
dimensionality of the vector
    xTrain = np.zeros(( totalNumGames, numFeatures + 1))
    yTrain = np.zeros(( totalNumGames ))
    indexCounter = 0
    for year in years:
        team_vectors = createSeasonDict(year)
        season = reg_season_compact_pd[reg_season_compact_pd['Season'] == year]
        numGamesInSeason = len(season.index)
        tourney = tourney_compact_pd[tourney_compact_pd['Season'] == year]
        numGamesInSeason += len(tourney.index)
        xTrainSeason = np.zeros(( numGamesInSeason, numFeatures + 1))
        yTrainSeason = np.zeros(( numGamesInSeason ))
        counter = 0
        for index, row in season.iterrows():
            w_team = row['WTeamID']
            w_vector = team_vectors[w_team]
            l_team = row['LTeamID']
            l_vector = team_vectors[l_team]
            diff = [a - b for a, b in zip(w_vector, l_vector)]
            home = getHomeStat(row['WLoc'])
            if (counter % 2 == 0):
                diff.append(home)
                xTrainSeason[counter] = diff
                yTrainSeason[counter] = 1
            else:
                diff.append(-home)
                xTrainSeason[counter] = [ -p for p in diff]
                yTrainSeason[counter] = 0
            counter += 1
        for index, row in tourney.iterrows():
            w_team = row['WTeamID']
            w_vector = team_vectors[w_team]
            l_team = row['LTeamID']
            l_vector = team_vectors[l_team]
            diff = [a - b for a, b in zip(w_vector, l_vector)]
            home = 0 #All tournament games are neutral
            if (counter % 2 == 0):
                diff.append(home)
                xTrainSeason[counter] = diff
                yTrainSeason[counter] = 1
            else:
                diff.append(-home)
                xTrainSeason[counter] = [ -p for p in diff]
                yTrainSeason[counter] = 0
            counter += 1
        xTrain[indexCounter:numGamesInSeason+indexCounter] = xTrainSeason
        yTrain[indexCounter:numGamesInSeason+indexCounter] = yTrainSeason
        indexCounter += numGamesInSeason
        print ('Finished year:', year)
        if (year in saveYears):
            np.save('Data/PrecomputedMatrices/TeamVectors/' + str(year) + 'TeamVectors', team_vectors)
    return xTrain, yTrain

def createAndSave(years, saveYears):
    xTrain, yTrain = createTrainingSet(years, saveYears)
    print ("Shape of xTrain:", xTrain.shape)
    print ("Shape of yTrain:", yTrain.shape)
    np.save('Data/PrecomputedMatrices/xTrain', xTrain)
    np.save('Data/PrecomputedMatrices/yTrain', yTrain)

############################ CREATE TRAINING SET ############################

# In case you want to run with Python 2
try:
    input = raw_input
except NameError:
    pass
```

```
endYear = int(input('What year do you have data until?\n'))

years = range(1993,endYear + 1)
# Saves the team vectors for the following years
saveYears = range(endYear - 4,endYear + 1)
if os.path.exists("Data/PrecomputedMatrices/xTrain.npy") and os.path.exists("Data/PrecomputedMatrices/
yTrain.npy"):
    print ('There is already a precomputed xTrain and yTrain.')
    response = input('Do you want to remove these files and create a new training set? (y/n) ')
    if (response == 'y'):
        os.remove("Data/PrecomputedMatrices/xTrain.npy")
        os.remove("Data/PrecomputedMatrices/yTrain.npy")
        createAndSave(years, saveYears)
    else:
        print ('Okay, going to exit now.')
else:
    createAndSave(years, saveYears)
```