```
# Format:
# 1) Imports
# 2) Load Training Set and CSV Files
# 3) Train Model
# 4) Test Model
# 5) Create Kaggle Submission

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

from __future__ import division
import sklearn
import pandas as pd
import numpy as np
import collections
import os.path
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.svm import SVC
from sklearn import linear_model
from sklearn import tree
from sklearn.model_selection import cross_val_score
from keras.utils import np_utils
from sklearn.neighbors import KNeighborsClassifier
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
import sys
from sklearn.ensemble import GradientBoostingRegressor
import math
import csv
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import classification_report
from sklearn.calibration import CalibratedClassifierCV
import urllib
from sklearn.svm import LinearSVC
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
from datetime import datetime
import random

############################## LOAD TRAINING SET ##############################

if os.path.exists("Data/PrecomputedMatrices/xTrain.npy") and os.path.exists("Data/PrecomputedMatrices/
yTrain.npy"):
    xTrain = np.load("Data/PrecomputedMatrices/xTrain.npy")
    yTrain = np.load("Data/PrecomputedMatrices/yTrain.npy")
    print ("Shape of xTrain:", xTrain.shape)
    print ("Shape of yTrain:", yTrain.shape)
else:
    print ('We need a training set! Run dataPreprocessing.py')
    sys.exit()
```

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

curYear = int(input('What year are these predictions for?\n'))

############################## LOAD CSV FILES ##############################

sample_sub_pd = pd.read_csv('Data/KaggleData/SampleSubmissionStage1.csv')
sample_sub_pd2 = pd.read_csv('Data/KaggleData/SampleSubmissionStage2.csv')
teams_pd = pd.read_csv('Data/KaggleData/Teams.csv')

############################## TRAIN MODEL ##############################

model = GradientBoostingRegressor(n_estimators=100, max_depth=5)

categories=['Wins','PPG','PPGA','PowerConf','3PG', 'APG','TOP','Conference Champ','Tourney
Conference Champ',
        'Seed','SOS','SRS', 'RPG', 'SPG', 'Tourney Appearances','National Championships','Location']
accuracy=[]
numTrials = 1

for i in range(numTrials):
   X_train, X_test, Y_train, Y_test = train_test_split(xTrain, yTrain)
   startTime = datetime.now() # For some timing stuff
   results = model.fit(X_train, Y_train)
   preds = model.predict(X_test)

   preds[preds < .5] = 0
   preds[preds >= .5] = 1
   localAccuracy = np.mean(preds == Y_test)
   accuracy.append(localAccuracy)
   print ("Finished run #" + str(i) + ". Accuracy = " + str(localAccuracy))
   print ("Time taken: " + str(datetime.now() - startTime))
if numTrials != 0:
   print ("The average accuracy is", sum(accuracy)/len(accuracy))

############################## TEST MODEL ##############################

def predictGame(team_1_vector, team_2_vector, home, modelUsed):
   diff = [a - b for a, b in zip(team_1_vector, team_2_vector)]
   diff.append(home)
   if hasattr(modelUsed, 'predict_proba'):
      return modelUsed.predict_proba([diff])[0][1]
   elif hasattr(modelUsed, 'predict'):
      return modelUsed.predict([diff])[0]
   else:
      raise AttributeError("Model does not have expected prediction method")

############################## CREATE KAGGLE SUBMISSION
##############################

def loadTeamVectors(years):
```

```
    listDictionaries = []
    for year in years:
        curVectors = np.load("Data/PrecomputedMatrices/TeamVectors/" + str(year) + "TeamVectors.npy",
mmap_mode = None, allow_pickle=True).item()
        listDictionaries.append(curVectors)
    return listDictionaries

def createPrediction(stage2 = False):
    if stage2:
        years = [curYear]
        localPd = sample_sub_pd2
    else:
        # The years that we want to predict for
        years = range(curYear - 4,curYear)
        localPd = sample_sub_pd

    if os.path.exists("result.csv"):
        os.remove("result.csv")
    listDictionaries = loadTeamVectors(years)
    print ("Loaded the team vectors")
    results = [[0 for x in range(2)] for x in range(len(localPd.index))]

    predictionModel = GradientBoostingRegressor(n_estimators=100, max_depth=5)
    predictionModel.fit(xTrain, yTrain)

    for index, row in localPd.iterrows():
        matchupId = row['ID']
        year = int(matchupId[0:4])
        teamVectors = listDictionaries[year - years[0]] ##here
        team1Id = int(matchupId[5:9])
        team2Id = int(matchupId[10:14])
        team1Vector = teamVectors[team1Id]
        team2Vector = teamVectors[team2Id]
        pred1 = predictGame(team1Vector, team2Vector, 0, predictionModel)
        pred = pred1.clip(0.,1.)
        results[index][0] = matchupId
        results[index][1] = pred
    results = pd.np.array(results)
    firstRow = [[0 for x in range(2)] for x in range(1)]
    firstRow[0][0] = 'ID'
    firstRow[0][1] = 'Pred'
    with open("result.csv", "w") as f:
        writer = csv.writer(f)
        writer.writerows(firstRow)
        writer.writerows(results)

#createPrediction()
createPrediction(stage2=True)

############################## PREDICTING THIS YEAR'S BRACKET
##############################

def trainModel():
    model = GradientBoostingRegressor(n_estimators=100, max_depth=5)
    model.fit(xTrain, yTrain)
```

```
    return model

def randomWinner(team1, team2, modelUsed, numTrials=10):
    year = [curYear]
    teamVectors = loadTeamVectors(year)[0]
    team1Vector = teamVectors[int(teams_pd[teams_pd['TeamName'] == team1].values[0][0])]
    team2Vector = teamVectors[int(teams_pd[teams_pd['TeamName'] == team2].values[0][0])]
    prediction = predictGame(team1Vector, team2Vector, 0, modelUsed)
    team1Wins = 0
    for i in range(numTrials):
        if (prediction > random.random()):
            team1Wins = team1Wins + 1
        print ("{0} Won {1} times".format(team1, team1Wins))


def findWinner(team1, team2, modelUsed):
    year = [curYear]
    teamVectors = loadTeamVectors(year)[0]
    team1Vector = teamVectors[int(teams_pd[teams_pd['TeamName'] == team1].values[0][0])]
    team2Vector = teamVectors[int(teams_pd[teams_pd['TeamName'] == team2].values[0][0])]
    prediction = predictGame(team1Vector, team2Vector, 0, modelUsed)
    if (prediction < 0.5):
        print ("Probability that {0} wins: {1}".format(team2, 1 - prediction))
    else:
        print ("Probability that {0} wins: {1}".format(team1, prediction))

def RnPWinner(team1, team2, modelUsed):
    year = [curYear]
    teamVectors = loadTeamVectors(year)[0]
    team1Vector = teamVectors[int(teams_pd[teams_pd['TeamName'] == team1].values[0][0])]
    team2Vector = teamVectors[int(teams_pd[teams_pd['TeamName'] == team2].values[0][0])]
    prediction = predictGame(team1Vector, team2Vector, 0, modelUsed)
    if (prediction < 0.5):
        print(team2)
        return team2
    else:
        print(team1)
        return team1

teams2019 = ['Duke', 'NC Central', 'VA Commonwealth', 'UCF', 'Mississippi St', 'Liberty',
             'Virginia Tech', 'St Louis', 'Maryland', 'Belmont', 'LSU', 'Yale', 'Louisville',
            'Minnesota', 'Michigan St', 'Bradley', 'Gonzaga', 'F Dickinson', 'Syracuse', 'Baylor',
            'Marquette', 'Murray St', 'Florida St', 'Vermont', 'Buffalo', 'Arizona St', 'Texas Tech',
            'N Kentucky', 'Nevada', 'Florida', 'Michigan', 'Montana','Virginia', 'Gardner Webb',
             'Mississippi', 'Oklahoma','Wisconsin', 'Oregon','Kansas St', 'UC Irvine','Villanova',
             'St Mary\'s CA', 'Purdue', 'Old Dominion','Cincinnati', 'Iowa','Colgate', 'Tennessee',
            'North Carolina', 'Iona','Utah St', 'Washington','Auburn', 'New Mexico St','Kansas',
             'Northwestern','Iowa St', 'Ohio St','Houston', 'Georgia St','Wofford', 'Seton Hall',
             'Kentucky', 'Abilene Chr']

def Bracket (teams, modelUsed):
    teamsv2=[]
    print('round of 64')
    for i in range(0,64,2):
        teamsv2.append(RnPWinner(teams[i], teams[i+1], modelUsed))
```

```
    teamsv3=[]
    print('\n\nround of 32')
    for i in range(0,32,2):
        teamsv3.append(RnPWinner(teamsv2[i], teamsv2[i+1], modelUsed))
    teamsv4=[]
    print('\n\nsweet 16')
    for i in range(0,16,2):
        teamsv4.append(RnPWinner(teamsv3[i], teamsv3[i+1], modelUsed))
    teamsv5=[]
    print('\n\nelite 8')
    for i in range(0,8,2):
        teamsv5.append(RnPWinner(teamsv4[i], teamsv4[i+1], modelUsed))
    teamsv6=[]
    print('\n\nfinal 4')
    for i in range(0,4,2):
        teamsv6.append(RnPWinner(teamsv5[i], teamsv5[i+1], modelUsed))
    teamsv7=[]
    print('\n\nchampionship game')
    for i in range(0,2,2):
        teamsv7.append(RnPWinner(teamsv6[i], teamsv6[i+1], modelUsed))
trainedModel = trainModel()
Bracket(teams2019, trainedModel)


'''

# First round games in the East for example

findWinner('Duke', 'NC Central', trainedModel)
findWinner('VA Commonwealth', 'UCF', trainedModel)
findWinner('Mississippi St', 'Liberty', trainedModel)
findWinner('Virginia Tech', 'St Louis', trainedModel)
findWinner('Maryland', 'Belmont', trainedModel)
findWinner('LSU', 'Yale', trainedModel)
findWinner('Louisville', 'Minnesota', trainedModel)
findWinner('Michigan St', 'Bradley', trainedModel)

# First round games in the west for example

findWinner('Gonzaga', 'F Dickinson', trainedModel)
findWinner('Syracuse', 'Baylor', trainedModel)
findWinner('Marquette', 'Murray St', trainedModel)
findWinner('Florida St', 'Vermont', trainedModel)
findWinner('Buffalo', 'Arizona St', trainedModel)
findWinner('Texas Tech', 'N Kentucky', trainedModel)
findWinner('Nevada', 'Florida', trainedModel)
findWinner('Michigan', 'Montana', trainedModel)

# First round games in the South for example

findWinner('Virginia', 'Gardner Webb', trainedModel)
findWinner('Mississippi', 'Oklahoma', trainedModel)
findWinner('Wisconsin', 'Oregon', trainedModel)
findWinner('Kansas St', 'UC Irvine', trainedModel)
findWinner('Villanova', 'St Mary\'s CA', trainedModel)
```

```
findWinner('Purdue', 'Old Dominion', trainedModel)
findWinner('Cincinnati', 'Iowa', trainedModel)
findWinner('Colgate', 'Tennessee', trainedModel)

# First round games in the midwest for example

findWinner('North Carolina', 'Iona', trainedModel)
findWinner('Utah St', 'Washington', trainedModel)
findWinner('Auburn', 'New Mexico St', trainedModel)
findWinner('Kansas', 'Northwestern', trainedModel)
findWinner('Iowa St', 'Ohio St', trainedModel)
findWinner('Houston', 'Georgia St', trainedModel)
findWinner('Wofford', 'Seton Hall', trainedModel)
findWinner('Kentucky', 'Abilene Chr', trainedModel)
'''
```