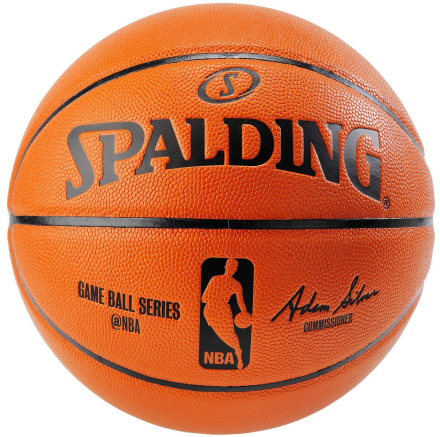
NCAA Bracket Predictor

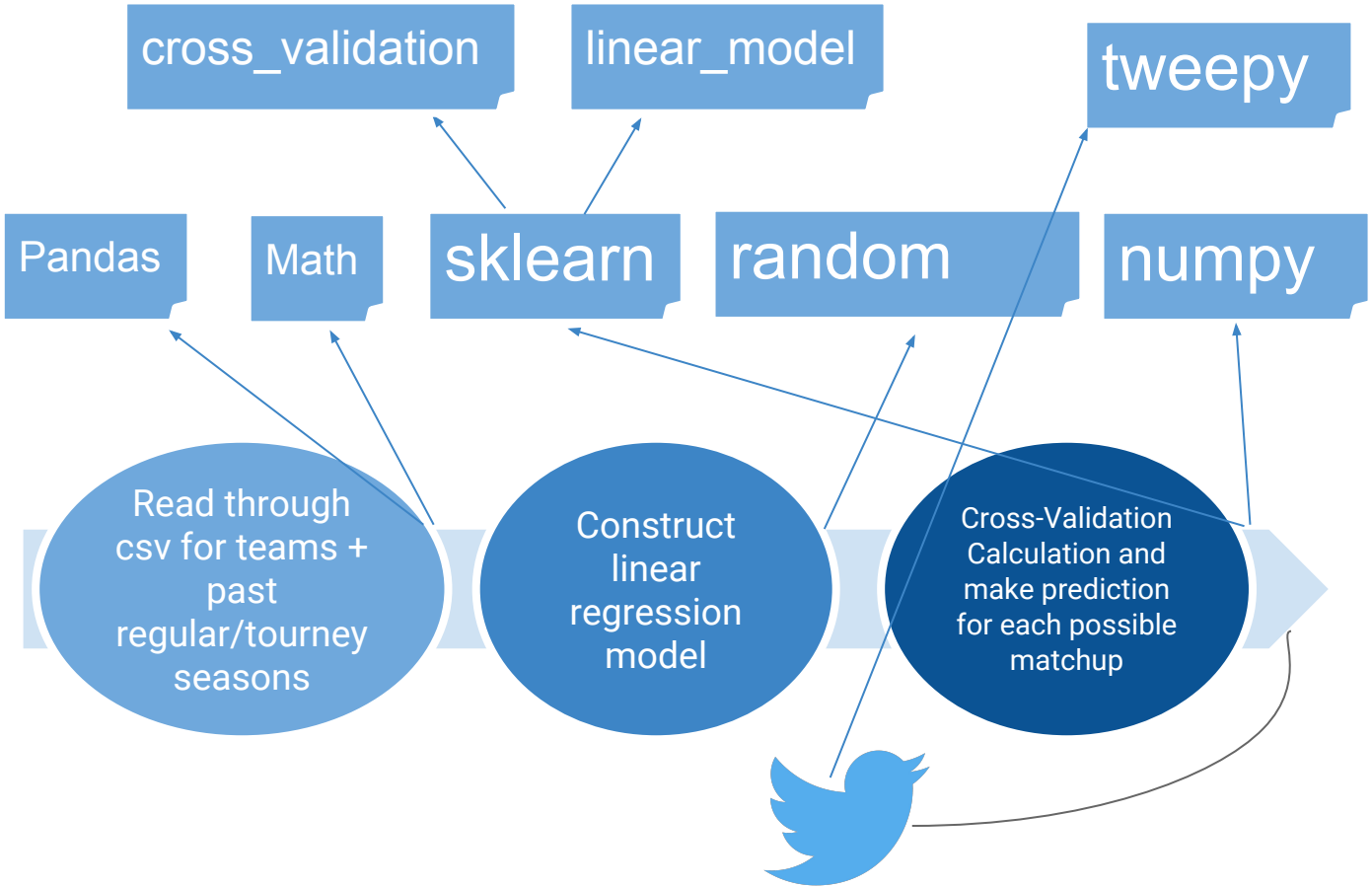
By Abhay Varshney

Introduction



- Calculate which team would win in NCAA basketball tournament
- Use ELO Ranking algorithm to compare 2 teams and figure out who would win
- Make Prediction based of ranking algorithm
- Acquire Tweets from Tweepy
- Respond with prediction based off tweet

Basic Process of Code



General Format - Data

(Kaggle)

REGULARSEASON.csv

Season	Daynum	Wteam	Wscore	Lteam	Lscore	Wloc	Numot	Wfgm	Wfga	Wfgm3	Wfga3	Wftm	Wfta	Wor	Wdr	Wast	Wto	Wstl	Wblk	Wpf	Lfgm	Lfga	Lfgm3	Lfga3	Lftm	Lfta	Lor	Ldr	Last	Lto	Lstl	Lblk	Lpf
2003	10	1104	68	1328	62	N	0	27	58	3	14	11	18	14	24	13	23	7	1	22	22	53	2	10	16	22	10	22	8	18	9	2	20
2003	10	1272	70	1393	63	N	0	26	62	8	20	10	19	15	28	16	13	4	4	18	24	67	6	24	9	20	20	25	7	12	8	6	16
2003	11	1266	73	1437	61	N	0	24	58	8	18	17	29	17	26	15	10	5	2	25	22	73	3	26	14	23	31	22	9	12	2	5	23
2003	11	1296	56	1457	50	N	0	18	38	3	9	17	31	6	19	11	12	14	2	18	18	49	6	22	8	15	17	20	9	19	4	3	23
2003	11	1400	77	1208	71	N	0	30	61	6	14	11	13	17	22	12	14	4	4	20	24	62	6	16	17	27	21	15	12	10	7	1	14
2003	11	1458	81	1186	55	H	0	26	57	6	12	23	27	12	24	12	9	9	3	18	20	46	3	11	12	17	6	22	8	19	4	3	25
2003	12	1161	80	1236	62	H	0	23	55	2	8	32	39	13	18	14	17	11	1	25	19	41	4	15	20	28	9	21	11	30	10	4	28
2003	12	1186	75	1457	61	N	0	28	62	4	14	15	21	13	35	19	19	7	2	21	20	59	4	17	17	23	8	25	10	15	14	8	18
2003	12	1194	71	1156	66	N	0	28	58	5	11	10	18	9	22	9	17	9	2	23	24	52	6	18	12	27	13	26	13	25	8	2	18
2003	12	1458	84	1296	56	H	0	32	67	5	17	15	19	14	22	11	6	12	0	13	23	52	3	14	7	12	9	23	10	18	1	3	18
2003	13	1166	106	1426	50	H	0	41	69	15	25	9	13	15	29	21	11	10	6	16	17	52	4	11	12	17	8	15	8	17	7	3	15
2003	13	1202	74	1106	73	N	0	29	51	7	13	9	11	6	21	18	15	7	1	5	29	63	10	22	5	5	13	16	15	12	6	2	12
2003	13	1237	66	1135	65	N	0	26	66	5	19	9	13	21	23	15	17	12	3	17	24	56	6	19	11	17	14	21	17	18	8	4	13
2003	13	1323	76	1125	48	H	0	25	56	10	23	16	23	8	35	18	13	14	19	13	18	64	8	24	4	8	14	26	12	17	10	0	17
2003	14	1125	83	1135	77	N	1	30	70	11	31	12	15	15	18	22	16	18	5	21	28	60	4	15	17	27	20	27	17	24	7	7	19
2003	14	1156	78	1236	71	N	0	27	46	10	18	14	24	8	25	18	15	2	6	18	23	50	7	20	18	24	6	17	21	11	5	1	21
2003	14	1161	81	1194	56	H	0	22	48	5	12	32	43	16	32	13	24	5	3	19	21	65	2	13	12	19	16	18	10	16	15	3	36
2003	14	1166	82	1202	57	H	0	33	61	8	19	8	14	9	23	21	17	14	5	18	23	55	2	11	9	14	13	23	9	26	8	4	18
2003	14	1183	73	1129	59	A	0	29	59	9	19	6	8	8	21	17	15	10	1	19	21	51	3	14	14	18	12	22	10	23	4	5	14
2003	14	1314	85	1336	55	H	0	32	60	5	15	16	23	12	34	18	14	8	4	12	21	71	3	23	10	11	15	20	6	13	7	5	19
2003	14	1323	89	1237	45	H	0	34	62	8	16	13	21	12	34	25	11	9	13	16	18	74	4	15	5	11	27	22	10	17	6	3	23
2003	14	1353	60	1162	36	H	0	23	57	4	19	10	15	15	18	10	12	14	6	18	12	39	3	8	9	13	11	23	4	27	2	4	19
2003	14	1390	61	1131	57	H	0	20	53	6	27	15	24	15	24	9	14	4	2	17	19	54	3	17	16	18	11	22	7	14	7	0	24
2003	14	1426	59	1106	47	N	0	25	53	1	5	8	12	15	32	14	16	6	4	7	20	60	6	18	1	2	8	15	10	8	10	6	14
2003	14	1462	87	1389	48	H	0	35	70	6	19	11	16	17	27	18	7	13	3	10	20	63	7	20	1	4	16	18	11	19	4	3	16
2003	15	1161	77	1156	63	H	0	27	58	5	10	18	27	18	26	15	12	5	4	21	23	51	3	13	14	20	7	19	13	13	6	6	22
2003	15	1194	93	1236	86	N	2	33	75	9	20	18	33	17	26	14	14	17	0	21	27	58	14	31	18	22	9	33	20	26	6	7	25
2003	15	1196	76	1256	55	H	0	28	48	4	13	16	24	8	23	18	18	13	5	19	19	49	1	5	16	22	12	17	14	22	8	2	22
2003	15	1242	81	1221	57	H	0	28	53	3	5	22	37	13	29	12	19	10	3	21	18	52	7	21	14	21	8	22	11	20	9	3	27
2003	15	1422	84	1447	65	H	0	33	53	3	8	15	20	7	36	12	22	5	9	19	26	70	3	25	10	16	14	15	9	12	10	2	16
2003	16	1314	71	1353	67	H	0	27	57	5	14	12	19	14	30	19	20	8	5	13	27	66	9	22	4	11	12	19	10	15	11	4	20
2003	16	1390	63	1462	62	H	0	23	57	6	15	11	21	17	20	12	10	8	1	19	18	45	6	20	20	28	11	17	6	14	5	1	19
2003	17	1196	99	1183	65	H	0	39	73	11	28	10	22	16	31	32	14	10	14	19	22	66	4	11	17	24	15	23	15	17	9	3	19
2003	17	1304	68	1147	45	N	0	27	58	6	18	8	13	10	28	13	17	5	5	26	10	50	2	16	23	31	16	24	3	20	6	3	19
2003	18	1104	82	1106	56	H	0	24	49	10	20	24	34	12	26	13	13	7	3	14	19	55	8	21	10	12	11	18	8	15	6	2	22
2003	18	1113	59	1287	56	H	0	22	54	1	12	14	25	13	27	12	15	6	5	15	22	63	7	24	5	9	16	23	13	18	6	4	25
2003	18	1116	81	1238	44	H	0	25	65	4	18	27	48	29	48	12	27	11	8	29	13	62	2	16	16	32	8	21	4	19	8	4	31
2003	18	1117	83	1369	71	H	0	30	72	7	19	16	20	24	17	12	12	8	3	14	27	52	5	15	12	16	11	20	10	18	2	4	17
2003	18	1120	81	1459	63	H	0	32	54	3	11	14	23	9	27	14	19	13	9	15	21	59	10	24	11	13	13	18	13	22	9	0	18
2003	18	1122	81	1272	80	A	1	29	70	11	24	12	17	13	27	19	13	6	5	27	24	62	8	19	24	34	13	31	17	14	4	7	17
2003	18	1123	72	1240	64	N	0	24	52	7	19	17	22	11	21	11	14	2	3	16	23	49	3	14	15	21	9	20	12	17	6	2	23
2003	18	1133	95	1337	81	H	1	34	79	5	24	22	34	19	40	17	15	7	1	21	29	71	9	29	14	22	8	35	15	15	6	3	30

Teams.csv

Team_Id	Team_Name
1101	Abilene Chr
1102	Air Force
1103	Akron
1104	Alabama
1105	Alabama A&M
1106	Alabama St
1107	Albany NY
1108	Alcorn St
1109	Alliant Intl
1110	American Univ
1111	Appalachian St
1112	Arizona
1113	Arizona St
1114	Ark Little Rock

TourneyResults.csv

Season	Seed	Team
1985	W01	1207
1985	W02	1210
1985	W03	1228
1985	W04	1260
1985	W05	1374
1985	W06	1208
1985	W07	1393
1985	W08	1396
1985	W09	1439
1985	W10	1177

Code Walkthrough: Main Function

- Reads Season stats
-

```
    # obtain bball score results from csv (obtained from Kaggle)
    # initialize necessary variables
    season_data = pd.read_csv('my_data/RegularSeasonDetailedResults.csv')
    tourney_data = pd.read_csv('my_data/TourneyDetailedResults.csv')
    seeds = pd.read_csv('my_data/TourneySeeds.csv')
    frames = [season_data, tourney_data]
    all_data = pd.concat(frames)
    model = linear_model.LogisticRegression()
    team_rating = {}
    total_matchups = []
    teamsArr = {}
    csv_data = []

    # initalize 2d list
    for i in range(1985, 2018):
        team_rating[i] = {}
        team_stats[i] = {}

    # Begin analyzing season and create model
    college_basketball_samples, binary_correct = analyzeSeason(all_data.iterrows(), team_rating)

    print("Total samples: %d" % len(college_basketball_samples))

    # Calculate accuracy using cross-validation sklearn
    print("Cross-validation: %f" % cross_validation.cross_val_score(model, numpy.array(college_bas

    print("Fitting samples to Logistic Regression model.")
    model.fit(college_basketball_samples, binary_correct)
    setUpTourney()

    # convert data to .csv
    print("Converting results to csv.")
    for index, col in pd.read_csv('my_data/Teams.csv').iterrows():
        teamsArr[col['Team_Id']] = col['Team_Name']
    for matchup in total_matchups:
        values = matchup[0].split('_')
        csv_data.append([teamsArr[int(values[1])], teamsArr[int(values[2])], matchup[1]])

    with open('my_data/my_predictions.csv', 'w') as f:
        writer = csv.writer(f)
        writer.writerows(csv_data)

    print("Beginning Twitter communication...")
    beginTwitter()
```

SkLearn

- **Logistic Regression Model**
  - Used to predict who wins based off stats of previous NCAA games
  - log-odds of the probability of an event is a linear combination of independent or predictor variables
- **Cross-Validation**
  - Returns accuracy of prediction using results from previous stats using Cross-Validation
  - Training data → college_basketball_samples & binary_correct

Code Walkthrough - Analyze Season

```
def analyzeSeason(season_data, team_rating):
    print("Analyzing Season Data and computing rating based of ELO algorithm.")
    for index, column in season_data:
        isUsable = True
        myYear = column['Season']  # gives year
        if column['Wloc'] == 'H':  # home team gets 100 in ranking
            team_a_ranking = 100 + getRating(column['Wteam'], myYear, team_rating)
            team_b_ranking = getRating(column['Lteam'], myYear, team_rating)
        else:
            team_a_ranking = getRating(column['Wteam'], myYear, team_rating)
            team_b_ranking = 100 + getRating(column['Lteam'], myYear, team_rating)
        copy_team_a_ranking = [team_a_ranking]
        copy_team_b_ranking = [team_b_ranking]

        for field in stats_fields:
            team_a_stats = calculateStatistics(column['Wteam'], myYear, field)
            team_b_stats = calculateStatistics(column['Lteam'], myYear, field)
            if team_a_stats is 0 and team_b_stats is 0:
                isUsable = False  # can't use these stats
            else:
                copy_team_a_ranking.append(team_a_stats)
                copy_team_b_ranking.append(team_b_stats)

        if isUsable:
            combineSamples(copy_team_a_ranking, copy_team_b_ranking)

        if column['Wfta'] != 0 and column['Lfta'] != 0:
            update_stats(myYear, column['Wteam'], setField('W', column))
            update_stats(myYear, column['Lteam'], setField('L', column))

        winner_rank = getRating(column['Wteam'], myYear, team_rating)
        loser_rank = getRating(column['Lteam'], myYear, team_rating)
        odds = 1 / (1 + math.pow(10, ((winner_rank - loser_rank) * -1) / 400))

        team_rating[myYear][column['Wteam']] = round(winner_rank + (getRank(winner_rank) * (1 - odds)))
        team_rating[myYear][column['Lteam']] = (loser_rank - (round(winner_rank + (getRank(winner_rank) * (1 - odds))) - winner_rank))

    return college_basketball_samples, binary_correct
```

Code Walkthrough - Make Prediction

```
def makePrediction(team_a, team_b, model, year, features, team_rating):
    team1Rating = getRating(team_a, year, team_rating)
    team2Rating = getRating(team_b, year, team_rating)

    features.append(team1Rating)
    for stat in stats_fields:
        year_stats = calculateStatistics(team_a, year, stat)
        features.append(year_stats)

    features.append(team2Rating)
    for stat in stats_fields:
        year_stats = calculateStatistics(team_b, year, stat)
        features.append(year_stats)

    return model.predict_proba([features])
```

Code Walkthrough - Prediction + Connect Prediction with Team Name

```
def setUpTourney():
    # obtain tournament teams
    tourney_teams = []
    for index, col in seeds.iterrows():
        if col['Season'] == 2017:
            tourney_teams.append(col['Team'])

    # Build our prediction of every matchup.
    print("Predicting matchups.")
    tourney_teams.sort()
    for team_1 in tourney_teams:
        for team_2 in tourney_teams:
            if team_2 > team_1:
                # print("%s beats %s. Prediction accuracy: %f." % (team_2, team_1, prediction[0][0]))
                label = str(2017) + '_' + str(team_1) + '_' + str(team_2)
                total_matchups.append([label, makePrediction(team_1, team_2, model, 2017, [], team_rating)[0][0]])
```

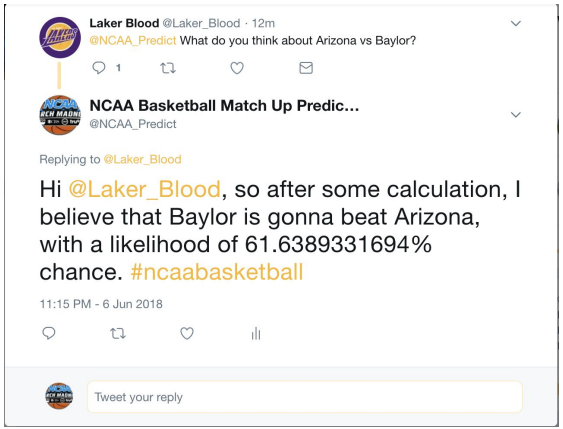
Sample Output of Prediction Results:

Arizona	**Arkansas**	**0.713786321751323**
Arizona	Baylor	0.616389331693955
Arizona	Bucknell	0.914968404875365
Arizona	Butler	0.642872527872693
Arizona	Cincinnati	0.6473955837288152
Arizona	Creighton	0.6984138358203831
Arizona	Dayton	0.7373784591102085
Arizona	Duke	0.5382249159463944
Arizona	ETSU	0.9005179129311065
Arizona	FL Gulf Coast	0.9194677902063007
Arizona	Florida	0.6750449208825784
Arizona	Florida St	0.6674151936265311
Arizona	Gonzaga	0.5382249159463944
Arizona	Iona	0.9094318095748143
Arizona	Iowa St	0.5987265113966642
Arizona	Jacksonville St	0.965743528154061
Arizona	Kansas	0.45813488371668154
Arizona	Kansas St	0.7650494050237683
Arizona	Kent	0.8996294332660116
Arizona	Kentucky	0.5111318650466458
Arizona	Louisville	0.5856059575776493

Twitter Feature

- Connect with Twitter
- Wait for response...
  - Once a response is received, read the tweet
  - Extract team names
- Using team names, read .csv file and look for prediction %
- Reply to the tweet using prediction %

Twitter Sample Results:



Accuracy:

~70%

Demo!

THANKS!