In [1]:
```
# Imports
# Standard
import pandas as pd
import numpy as np

# Preprocessing and crossvalidation
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV

# Models
from sklearn.linear_model import LogisticRegressionCV
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# Metrics
from sklearn.metrics import log_loss
```

In [2]:
```
# Read in training data
train = pd.read_csv('./data/training_data_2003-21.csv')
train = train.drop(['Unnamed: 0'], axis=1)
print(train.shape)
train.head(5)
```

```
(2274, 45)
```

Out[2]:

	Opp_Score	Opp_FGM	Opp_FGA	Opp_FGM3	Opp_FGA3	Opp_FTM	Opp_FTA	Opp_O
0	79.703704	28.074074	61.407407	5.222222	15.185185	18.333333	24.814815	14.00000
1	78.800000	28.533333	58.500000	7.400000	18.566667	14.333333	20.233333	10.36666
2	73.636364	26.272727	55.696970	7.454545	20.515152	13.636364	19.606061	11.18181
3	73.933333	26.533333	54.800000	5.200000	14.200000	15.666667	20.833333	11.10000
4	68.093750	23.562500	52.843750	5.312500	15.406250	15.656250	22.531250	12.31250

5 rows × 45 columns

In [3]:

```
# Scale and run PCA to reduce number of features
scaler = StandardScaler()
X = scaler.fit_transform(train.drop(['Win'], axis=1))
y = train.Win

n = 9 # 90% variance explained at 18, 99.9% variance explained at 35
pca = PCA(n_components=n)
pcaX = pca.fit_transform(X)
print('Variance explained by %d components: %.4f' %(n, pca.explained_variance
```

```
Variance explained by 9 components: 70.3998
```

In [4]:

```
# Fit the models on PCA training data

###### Logistic Regression ######
cv = 5                      # 5-fold cross validation
Cs = 10**np.linspace(-5,5) # constants for cost function
n_CPU = 8                   # number of CPUs to use for CV

lgr = LogisticRegressionCV(Cs=Cs, cv=cv, n_jobs=n_CPU)
lgr.fit(pcaX,y)
print('Logistic regression (lgr) score: %.6f' %(lgr.score(pcaX,y)))

###### Support Vector Machine ######
parameters = {'C':10**np.linspace(-5,5)}

svc = SVC(probability=True)
svc_CV = GridSearchCV(svc, parameters)
svc_CV.fit(pcaX,y)
svc_best = svc_CV.best_estimator_

print('Support vector classifier (svc_best) score: %.6f' %(svc_best.score(pca

###### Decision Tree ######
parameters = {'max_depth':range(3,20)}

dtc = DecisionTreeClassifier()
dtc_CV = GridSearchCV(dtc, parameters)
dtc_CV.fit(pcaX,y)
dtc_best = dtc_CV.best_estimator_

print('Decision tree classifier (dtc_best) score: %.6f' %(dtc_best.score(pcaX

###### Random Forest ######
parameters = {'max_depth':range(3,20)}

rf = RandomForestClassifier(n_jobs=n_CPU)
rf_CV = GridSearchCV(rf, parameters)
rf_CV.fit(pcaX,y)
rf_best = rf_CV.best_estimator_

print('Random forest (rf_best) score: %.6f' %(rf_best.score(pcaX,y)))
```

```
Logistic regression (lgr) score: 0.707564
Support vector classifier (svc_best) score: 0.719437
Decision tree classifier (dtc_best) score: 0.705365
Random forest (rf_best) score: 0.866315
```

In [5]:
```
# Calculate log loss for each model on training data
models = [lgr, svc_best, dtc_best, rf_best]
names = ['Logistic regression', 'SVC', 'Decision tree', 'Random forest']

print('Log loss on training data:')
for m,n in zip(models,names):
    preds = m.predict_proba(pcaX)
    loss = log_loss(y, preds)
    print('\t%s: %.4f' %(n, loss))
```

```
Log loss on training data:
        Logistic regression: 0.5521
        SVC: 0.5450
        Decision tree: 0.5718
        Random forest: 0.4059
```

In [6]:
```
# Some functions for predicting games
def make_game(team1_name, team2_name, data):

    team1 = data[data.TeamName == team1_name].reset_index(drop=True)
    team2 = data[data.TeamName == team2_name].reset_index(drop=True)

    t1 = team1.drop('TeamName', axis=1)
    t2 = team2.drop('TeamName', axis=1)

    opp_cols = {}
    for col in t2.columns:
        opp_cols[col] = 'Opp_' + col

    t2 = t2.rename(columns=opp_cols)

    game = pd.concat([t2,t1], axis=1)

    return game

def get_seeds(team1_name, team2_name, data):

    team1 = data[data.TeamName == team1_name].reset_index(drop=True)
    team2 = data[data.TeamName == team2_name].reset_index(drop=True)

    return team1.Seed, team2.Seed
```

In [7]:
```
# Read in Tournament Seeds to add to testing data
seeds = pd.read_csv('./data/MNCAATourneySeeds.csv')

seeds = seeds[seeds.Season >= 2003]

seeds_clean = []
for seed in seeds.Seed.tolist():
    seeds_clean.append(int(seed.strip('WXYZab')))


seeds['SeedsClean'] = seeds_clean
seeds = seeds.drop('Seed', axis=1).rename(columns={'SeedsClean' : 'Seed'})
```

In [8]:
```
# Read in testing data for 2022
test = pd.read_csv('./data/testing_data_2022.csv')
test = test.drop(['Unnamed: 0'], axis=1)

# Add in Seed data to testing data
yr = 2022
seeds_yr = seeds[seeds.Season == yr]
test = test.merge(seeds_yr.drop('Season', axis=1),
                  left_on='TeamID', right_on='TeamID')
test = test.drop(['Season','TeamID','WLK','POM','MAS','SAG'], axis=1)
test = test[[c for c in test if c not in ['Seed', 'AvgRank']]
       + ['Seed', 'AvgRank']]

test.head(5)
```

Out[8]:

	Score	FGM	FGA	FGM3	FGA3	FTM	FTA	OR
0	79.968750	27.656250	62.750000	9.281250	30.093750	15.375000	21.062500	11.625000
1	84.558824	30.441176	61.382353	7.764706	21.911765	15.911765	21.558824	10.441176
2	76.939394	26.636364	60.606061	6.363636	20.757576	17.303030	22.969697	9.696970
3	78.718750	27.937500	63.562500	8.125000	25.375000	14.718750	20.093750	10.218750
4	68.121212	24.242424	54.424242	7.181818	21.030303	12.454545	19.151515	8.787879

5 rows × 23 columns

In [9]:
```
# Input arrays for 2022 bracket
round_of_64 = [
    ['Gonzaga',        'Georgia St'],
    ['Boise St',    'Memphis'],
    ['Connecticut',    'New Mexico St'],
    ['Arkansas',       'Vermont'],
    ['Alabama',        'Notre Dame'],
    ['Texas Tech',     'Montana St'],
```

```
    ['Michigan St',    'Davidson'],
    ['Duke',           'CS Fullerton'],
    ['Baylor',         'Norfolk St'],
    ['North Carolina', 'Marquette'],
    ["St Mary's CA",    'Indiana'],
    ['UCLA',           'Akron'],
    ['Texas',          'Virginia Tech'],
    ['Purdue',         'Yale'],
    ['Murray St',      'San Francisco'],
    ['Kentucky',       "St Peter's"],
    ['Arizona',        'Wright St'],
    ['Seton Hall',     'TCU'],
    ['Houston',        'UAB'],
    ['Illinois',       'Chattanooga'],
    ['Colorado St',    'Michigan'],
    ['Tennessee',      'Longwood'],
    ['Ohio St',        'Loyola-Chicago'],
    ['Villanova',      'Delaware'],
    ['Kansas',         'TX Southern'],
    ['San Diego St',   'Creighton'],
    ['Iowa',           'Richmond'],
    ['Providence',     'S Dakota St'],
    ['LSU',            'Iowa St'],
    ['Wisconsin',      'Colgate'],
    ['USC',            'Miami FL'],
    ['Auburn',         'Jacksonville St']
]

round_of_32 = [
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B']
]

sweet_16 = [
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B'],
```

```
    ['A','B'],
    ['A','B'],
    ['A','B']
]

elite_8 = [
    ['A','B'],
    ['A','B'],
    ['A','B'],
    ['A','B']
]

final_4 = [
    ['A','B'],
    ['A','B']
]

ship = ['A','B']
```

In [10]:

```
# Set some choices for predictions
show_prob = True
model = rf_best
```

In [11]:

```
# Predict Round of 64
# Generate Round of 64 games
games_64 = pd.DataFrame()
for g,teams in enumerate(round_of_64):
    games_64 = pd.concat([games_64, make_game(teams[0],teams[1],test)])

X = scaler.transform(games_64)
pcaX = pca.transform(X)

if show_prob:
    probs = model.predict_proba(pcaX)[:,1]
preds = model.predict(pcaX)

print('-- Round of 64 --')
i = 0
j = 0
regions = ['West','East','South','Midwest']
for g,teams in enumerate(round_of_64):
    if g % 8 == 0:
        print('\n%s Region:' %(regions[i]))
        i += 1

    s1, s2 = get_seeds(teams[0], teams[1], test)

    if preds[g] == 1:

        if g % 2 == 0:
            round_of_32[j][0] = teams[0]
        elif g % 2 == 1:
            round_of_32[j][1] = teams[0]
            j += 1

        if show_prob:
            print('%s (%d) beats %s (%d) with probability: %.4f'
                  %(teams[0], s1, teams[1], s2, probs[g]))
        else:
            print('%s (%d) beats %s (%d)' %(teams[0],s1,teams[1],s2))
    else:

        if g % 2 == 0:
            round_of_32[j][0] = teams[1]
        elif g % 2 == 1:
            round_of_32[j][1] = teams[1]
            j += 1

        if show_prob:
            print('%s (%d) beats %s (%d) with probability: %.4f'
                  %(teams[1], s2, teams[0], s1, 1-probs[g]))
        else:
            print('%s (%d) beats %s (%d)' %(teams[1],s2,teams[0],s1))
```

```
-- Round of 64 --

West Region:
Gonzaga (1) beats Georgia St (16) with probability: 0.9136
Memphis (9) beats Boise St (8) with probability: 0.5748
Connecticut (5) beats New Mexico St (12) with probability: 0.8695
Vermont (13) beats Arkansas (4) with probability: 0.6328
Alabama (6) beats Notre Dame (11) with probability: 0.6033
Texas Tech (3) beats Montana St (14) with probability: 0.8306
Davidson (10) beats Michigan St (7) with probability: 0.5477
Duke (2) beats CS Fullerton (15) with probability: 0.8986

East Region:
Baylor (1) beats Norfolk St (16) with probability: 0.8364
North Carolina (8) beats Marquette (9) with probability: 0.6054
St Mary's CA (5) beats Indiana (12) with probability: 0.6506
UCLA (4) beats Akron (13) with probability: 0.8845
Virginia Tech (11) beats Texas (6) with probability: 0.6212
Purdue (3) beats Yale (14) with probability: 0.8520
Murray St (7) beats San Francisco (10) with probability: 0.5594
Kentucky (2) beats St Peter's (15) with probability: 0.8523

South Region:
Arizona (1) beats Wright St (16) with probability: 0.9169
Seton Hall (8) beats TCU (9) with probability: 0.6995
Houston (5) beats UAB (12) with probability: 0.8010
Illinois (4) beats Chattanooga (13) with probability: 0.7614
Colorado St (6) beats Michigan (11) with probability: 0.5489
Tennessee (3) beats Longwood (14) with probability: 0.8732
Loyola-Chicago (10) beats Ohio St (7) with probability: 0.5711
Villanova (2) beats Delaware (15) with probability: 0.7957

Midwest Region:
Kansas (1) beats TX Southern (16) with probability: 0.8968
Creighton (9) beats San Diego St (8) with probability: 0.5911
Iowa (5) beats Richmond (12) with probability: 0.8340
Providence (4) beats S Dakota St (13) with probability: 0.5885
LSU (6) beats Iowa St (11) with probability: 0.6494
Colgate (14) beats Wisconsin (3) with probability: 0.5606
USC (7) beats Miami FL (10) with probability: 0.6789
Auburn (2) beats Jacksonville St (15) with probability: 0.7898
```

In [12]:

```
# Predict Round of 32
# Generate Round of 32 games with results from Round of 64
games_32 = pd.DataFrame()
for g,teams in enumerate(round_of_32):
    games_32 = pd.concat([games_32, make_game(teams[0],teams[1],test)])

X = scaler.transform(games_32)
pcaX = pca.transform(X)

if show_prob:
    probs = model.predict_proba(pcaX)[:,1]
preds = model.predict(pcaX)


print('-- Round of 32 --')
i = 0
j = 0
regions = ['West','East','South','Midwest']
for g,teams in enumerate(round_of_32):
    if g % 4 == 0:
        print('\n%s Region:' %(regions[i]))
        i += 1

    s1, s2 = get_seeds(teams[0], teams[1], test)

    if preds[g] == 1:

        if g % 2 == 0:
            sweet_16[j][0] = teams[0]
        elif g % 2 == 1:
            sweet_16[j][1] = teams[0]
            j += 1

        if show_prob:
            print('%s (%d) beats %s (%d) with probability: %.4f'
                  %(teams[0], s1, teams[1], s2, probs[g]))
        else:
            print('%s (%d) beats %s (%d)' %(teams[0],s1,teams[1],s2))
    else:

        if g % 2 == 0:
            sweet_16[j][0] = teams[1]
        elif g % 2 == 1:
            sweet_16[j][1] = teams[1]
            j += 1

        if show_prob:
            print('%s (%d) beats %s (%d) with probability: %.4f'
                  %(teams[1], s2, teams[0], s1, 1-probs[g]))
        else:
            print('%s (%d) beats %s (%d)' %(teams[1],s2,teams[0],s1))
```

```
-- Round of 32 --

West Region:
Gonzaga (1) beats Memphis (9) with probability: 0.8822
Vermont (13) beats Connecticut (5) with probability: 0.5609
Texas Tech (3) beats Alabama (6) with probability: 0.7593
Duke (2) beats Davidson (10) with probability: 0.8124

East Region:
Baylor (1) beats North Carolina (8) with probability: 0.7102
UCLA (4) beats St Mary's CA (5) with probability: 0.6068
Purdue (3) beats Virginia Tech (11) with probability: 0.6128
Kentucky (2) beats Murray St (7) with probability: 0.6255

South Region:
Arizona (1) beats Seton Hall (8) with probability: 0.8361
Houston (5) beats Illinois (4) with probability: 0.7156
Tennessee (3) beats Colorado St (6) with probability: 0.5811
Villanova (2) beats Loyola-Chicago (10) with probability: 0.6037

Midwest Region:
Kansas (1) beats Creighton (9) with probability: 0.7709
Iowa (5) beats Providence (4) with probability: 0.5821
LSU (6) beats Colgate (14) with probability: 0.5276
Auburn (2) beats USC (7) with probability: 0.7791
```

In [13]:

```
# Predict Sweet 16
# Generate Sweet 16 games
games_16 = pd.DataFrame()
for g,teams in enumerate(sweet_16):
    games_16 = pd.concat([games_16, make_game(teams[0],teams[1],test)])


X = scaler.transform(games_16)
pcaX = pca.transform(X)

if show_prob:
    probs = model.predict_proba(pcaX)[:,1]
preds = model.predict(pcaX)


print('-- Sweet 16 --')
i = 0
j = 0
regions = ['West','East','South','Midwest']
for g,teams in enumerate(sweet_16):
    if g % 2 == 0:
        print('\n%s Region:' %(regions[i]))
        i += 1

    s1, s2 = get_seeds(teams[0], teams[1], test)

    if preds[g] == 1:
```

```
        if g % 2 == 0:
            elite_8[j][0] = teams[0]
        elif g % 2 == 1:
            elite_8[j][1] = teams[0]
            j += 1

        if show_prob:
            print('%s (%d) beats %s (%d) with probability: %.4f'
                  %(teams[0], s1, teams[1], s2, probs[g]))
        else:
            print('%s (%d) beats %s (%d)' %(teams[0],s1,teams[1],s2))
    else:

        if g % 2 == 0:
            elite_8[j][0] = teams[1]
        elif g % 2 == 1:
            elite_8[j][1] = teams[1]
            j += 1

        if show_prob:
            print('%s (%d) beats %s (%d) with probability: %.4f'
                  %(teams[1], s2, teams[0], s1, 1-probs[g]))
        else:
            print('%s (%d) beats %s (%d)' %(teams[1],s2,teams[0],s1))
```

```
-- Sweet 16 --

West Region:
Gonzaga (1) beats Vermont (13) with probability: 0.7571
Duke (2) beats Texas Tech (3) with probability: 0.5941

East Region:
Baylor (1) beats UCLA (4) with probability: 0.5606
Kentucky (2) beats Purdue (3) with probability: 0.7118

South Region:
Arizona (1) beats Houston (5) with probability: 0.5857
Tennessee (3) beats Villanova (2) with probability: 0.5191

Midwest Region:
Iowa (5) beats Kansas (1) with probability: 0.5846
Auburn (2) beats LSU (6) with probability: 0.7155
```

In [14]:

```
# Predict Elite 8
# Generate Elite 8 games
games_8 = pd.DataFrame()
for g,teams in enumerate(elite_8):
    games_8 = pd.concat([games_8, make_game(teams[0],teams[1],test)])


X = scaler.transform(games_8)
pcaX = pca.transform(X)
```

```
if show_prob:
    probs = model.predict_proba(pcaX)[:,1]
preds = model.predict(pcaX)

print('-- Elite 8 --')
i = 0
j = 0
regions = ['West','East','South','Midwest']
for g,teams in enumerate(elite_8):
    if g % 1 == 0:
        print('\n%s Region:' %(regions[i]))
        i += 1

    s1, s2 = get_seeds(teams[0], teams[1], test)

    if preds[g] == 1:

        if g % 2 == 0:
            final_4[j][0] = teams[0]
        elif g % 2 == 1:
            final_4[j][1] = teams[0]
            j += 1

        if show_prob:
            print('%s (%d) beats %s (%d) with probability: %.4f'
                  %(teams[0], s1, teams[1], s2, probs[g]))
        else:
            print('%s (%d) beats %s (%d)' %(teams[0],s1,teams[1],s2))
    else:

        if g % 2 == 0:
            final_4[j][0] = teams[1]
        elif g % 2 == 1:
            final_4[j][1] = teams[1]
            j += 1

        if show_prob:
            print('%s (%d) beats %s (%d) with probability: %.4f'
                  %(teams[1], s2, teams[0], s1, 1-probs[g]))
        else:
            print('%s (%d) beats %s (%d)' %(teams[1],s2,teams[0],s1))
```

```
-- Elite 8 --

West Region:
Gonzaga (1) beats Duke (2) with probability: 0.7290

East Region:
Kentucky (2) beats Baylor (1) with probability: 0.6168

South Region:
Arizona (1) beats Tennessee (3) with probability: 0.7209

Midwest Region:
Auburn (2) beats Iowa (5) with probability: 0.5350
```

In [15]:

```
# Predict Final Four
# Generate Final Four games
games_4 = pd.DataFrame()
for g,teams in enumerate(final_4):
    games_4 = pd.concat([games_4, make_game(teams[0],teams[1],test)])


X = scaler.transform(games_8)
pcaX = pca.transform(X)

if show_prob:
    probs = model.predict_proba(pcaX)[:,1]
preds = model.predict(pcaX)

print('-- Final 4 --')
i = 0
for g,teams in enumerate(final_4):

    s1, s2 = get_seeds(teams[0], teams[1], test)

    if preds[g] == 1:

        ship[i] = teams[0]

        if show_prob:
            print('%s (%d) beats %s (%d) with probability: %.4f'
                  %(teams[0], s1, teams[1], s2, probs[g]))
        else:
            print('%s (%d) beats %s (%d)' %(teams[0],s1,teams[1],s2))
    else:

        ship[i] = teams[1]

        if show_prob:
            print('%s (%d) beats %s (%d) with probability: %.4f'
                  %(teams[1], s2, teams[0], s1, 1-probs[g]))
        else:
            print('%s (%d) beats %s (%d)' %(teams[1],s2,teams[0],s1))


    i += 1
```

```
-- Final 4 --
Gonzaga (1) beats Kentucky (2) with probability: 0.7290
Auburn (2) beats Arizona (1) with probability: 0.6168
```

In [16]:
```
# Predict Championship
# Generate Championship game
champ = make_game(ship[0], ship[1], test)
champ

X = scaler.transform(champ)
pcaX = pca.transform(X)

if show_prob:
    probs = model.predict_proba(pcaX)[:,1]
preds = model.predict(pcaX)

teams = ship
s1, s2 = get_seeds(teams[0], teams[1], test)
g = 0

if preds == 1:
    if show_prob:
        print('%s (%d) beats %s (%d) with probability: %.4f'
              %(teams[0], s1, teams[1], s2, probs[g]))
    else:
        print('%s (%d) beats %s (%d)' %(teams[0],s1,teams[1],s2))
else:
    if show_prob:
        print('%s (%d) beats %s (%d) with probability: %.4f'
              %(teams[1], s2, teams[0], s1, 1-probs[g]))
    else:
        print('%s (%d) beats %s (%d)' %(teams[1],s2,teams[0],s1))
```

```
Gonzaga (1) beats Auburn (2) with probability: 0.7722
```

In [18]:
```
# Predict the championship score
# Average winning championship score from 2003 - 2021
# Average losing championship score from 2003 - 2021
t = pd.read_csv('./data/MNCAATourneyDetailedResults.csv')
t = t[t.DayNum == 154] # only look at championships
print('Average winning score: ', t.WScore.mean())
print('Average losing score: ', t.LScore.mean())
print('Total average score: ', t.WScore.mean() + t.LScore.mean())
```

```
Average winning score:  74.88888888888889
Average losing score:  66.27777777777777
Total average score:  141.16666666666666
```

In [ ]:
```

```