FIFA Prediction

Objective: Dev ML model to predict First, Second and Third Place for 2018 FIFA worldcup
Features: Kaggle historical data for past matches including friendly games, Eloranking by country
Purpose: Submissions for DBS internal competition (Due 25/06/2018)

1) Data prep

Process the history kaggle data from results.csv (1930 onwards due to limitation of elorating data)
Possible integration with elorating (Used custom javascript to crawl data from https://www.eloratings.net/ (https://www.eloratings.net/))

In [1]:
```
# import libraries for data manipulation
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# example making new class predictions for a classification problem
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout, TimeDistributed, AveragePooling1D, Flatten
from keras.utils import to_categorical
from keras.optimizers import Adam, RMSprop

# back up model graph
from keras.models import load_model

# using sklearn to have 1 liner cross validation
from sklearn.model_selection import train_test_split
```

```
c:\users\poryee\appdata\local\programs\python\python35\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversi
on of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as
`np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [2]:
```
# Read .csv files from kaggle
results = pd.read_csv('datasets/results.csv')
```

In [3]:
```
# observe results
results.head()
```

Out[3]:

	date	home_team	away_team	home_score	away_score	tournament	city	country
0	1872-11-30	Scotland	England	0	0	Friendly	Glasgow	Scotland
1	1873-03-08	England	Scotland	4	2	Friendly	London	England
2	1874-03-07	Scotland	England	2	1	Friendly	Glasgow	Scotland
3	1875-03-06	England	Scotland	2	2	Friendly	London	England
4	1876-03-04	Scotland	England	3	0	Friendly	Glasgow	Scotland

Through the exploration of data we need to find the absolute difference in score and the winning team
Append the corresponding results to the newly created columns [wining_team]
And finally, keep data of teams that make it to the group stage while dropping the rest

In [4]:
```
# Adding new column for winner of each match
winner = []
for i in range(len(results['home_team'])):
    if results['home_score'][i] > results['away_score'][i]:
        winner.append(results['home_team'][i])
    elif results['home_score'][i] < results['away_score'][i]:
        winner.append(results['away_team'][i])
    else:
        winner.append('Tie')
results['winning_team'] = winner

# Adding new column for goal difference in matches
results['goal_difference'] = np.absolute(results['home_score'] - results['away_score'])

# view new sample header
results.head()
```

Out[4]:

	date	home_team	away_team	home_score	away_score	tournament	city	country	winning_team	goal_difference
0	1872-11-30	Scotland	England	0	0	Friendly	Glasgow	Scotland	Tie	0
1	1873-03-08	England	Scotland	4	2	Friendly	London	England	England	2
2	1874-03-07	Scotland	England	2	1	Friendly	Glasgow	Scotland	Scotland	1
3	1875-03-06	England	Scotland	2	2	Friendly	London	England	Tie	0
4	1876-03-04	Scotland	England	3	0	Friendly	Glasgow	Scotland	Scotland	3

In [5]:
```
# scope current worldcup team qualifing teams
wc_teams = ['Australia', ' Iran', 'Japan', 'Korea Republic',
            'Saudi Arabia', 'Egypt', 'Morocco', 'Nigeria',
            'Senegal', 'Tunisia', 'Costa Rica', 'Mexico',
            'Panama', 'Argentina', 'Brazil', 'Colombia',
            'Peru', 'Uruguay', 'Belgium', 'Croatia',
            'Denmark', 'England', 'France', 'Germany',
            'Iceland', 'Poland', 'Portugal', 'Russia',
            'Serbia', 'Spain', 'Sweden', 'Switzerland']

# Filter the 'results' dataframe to show only teams in this years' world cup, from 1930 onwards
# we only care about teams that qualify
df_teams_home = results[results['home_team'].isin(wc_teams)]
df_teams_away = results[results['away_team'].isin(wc_teams)]
df_teams = pd.concat((df_teams_home, df_teams_away))
df_teams.drop_duplicates()
```

Out[5]:

	date	home_team	away_team	home_score	away_score	tournament	city	country	winning_team	goal_diffe
1	1873-03-08	England	Scotland	4	2	Friendly	London	England	England	2
3	1875-03-06	England	Scotland	2	2	Friendly	London	England	Tie	0
6	1877-03-03	England	Scotland	1	3	Friendly	London	England	Scotland	2
10	1879-01-18	England	Wales	2	1	Friendly	London	England	England	1
11	1879-04-05	England	Scotland	5	4	Friendly	London	England	England	1
16	1881-02-26	England	Wales	0	1	Friendly	Blackburn	England	Wales	1
17	1881-03-12	England	Scotland	1	6	Friendly	London	England	Scotland	5
24	1883-02-03	England	Wales	5	0	Friendly	London	England	England	5
25	1883-02-24	England	Northern Ireland	7	0	Friendly	Liverpool	England	England	7
26	1883-03-10	England	Scotland	2	3	Friendly	Sheffield	England	Scotland	1
35	1885-02-28	England	Northern Ireland	4	0	British Championship	Manchester	England	England	4
36	1885-03-14	England	Wales	1	1	British Championship	Blackburn	England	Tie	0
38	1885-03-21	England	Scotland	1	1	British Championship	London	England	Tie	0

	date	home_team	away_team	home_score	away_score	tournament	city	country	winning_team	goal_diffe
49	1887-02-05	England	Northern Ireland	7	0	British Championship	Sheffield	England	England	7
51	1887-02-26	England	Wales	4	0	British Championship	London	England	England	4
53	1887-03-19	England	Scotland	2	3	British Championship	Blackburn	England	Scotland	1
62	1889-02-23	England	Wales	4	1	British Championship	Stoke-on-Trent	England	England	3
63	1889-03-02	England	Northern Ireland	6	1	British Championship	Liverpool	England	England	5
65	1889-04-13	England	Scotland	2	3	British Championship	London	England	Scotland	1
75	1891-03-07	England	Wales	4	1	British Championship	Sunderland	England	England	3
76	1891-03-07	England	Northern Ireland	6	1	British Championship	Wolverhampton	England	England	5
85	1893-02-25	England	Northern Ireland	6	1	British Championship	Birmingham	England	England	5
86	1893-03-13	England	Wales	6	0	British Championship	Stoke-on-Trent	England	England	6
89	1893-04-01	England	Scotland	5	2	British Championship	Richmond	England	England	3
96	1895-03-09	England	Northern Ireland	9	0	British Championship	Derby	England	England	9
98	1895-03-18	England	Wales	1	1	British Championship	London	England	Tie	0
101	1895-04-06	England	Scotland	3	0	British Championship	Liverpool	England	England	3

	date	home_team	away_team	home_score	away_score	tournament	city	country	winning_team	goal_diffe
108	1897-02-20	England	Northern Ireland	6	0	British Championship	Nottingham	England	England	6
112	1897-03-29	England	Wales	4	0	British Championship	Sheffield	England	England	4
113	1897-04-03	England	Scotland	1	2	British Championship	London	England	Scotland	1
...	...	...	...	...	...	...	...	...	...	...
38644	2017-11-13	Bulgaria	Saudi Arabia	1	0	Friendly	Lisbon	Portugal	Bulgaria	1
38661	2017-11-14	Ireland	Denmark	1	5	FIFA World Cup qualification	Dublin	Ireland	Denmark	4
38663	2017-11-14	Hungary	Costa Rica	1	0	Friendly	Budapest	Hungary	Hungary	1
38667	2017-11-14	China	Colombia	0	4	Friendly	Chongqing	China	Colombia	4
38669	2017-11-14	Qatar	Iceland	1	1	Friendly	Doha	Qatar	Tie	0
38671	2017-11-14	Wales	Panama	1	1	Friendly	Cardiff	Wales	Tie	0
38673	2017-11-14	Austria	Uruguay	2	1	Friendly	Vienna	Austria	Austria	1
38722	2017-12-12	Korea DPR	Korea Republic	0	1	EAFF Championship	Chōfu	Japan	Korea Republic	1
38736	2017-12-22	Kuwait	Saudi Arabia	1	2	Friendly	Kuwait	Kuwait	Saudi Arabia	1
38740	2017-12-25	United Arab Emirates	Saudi Arabia	0	0	Friendly	Kuwait	Kuwait	Tie	0

	date	home_team	away_team	home_score	away_score	tournament	city	country	winning_team	goal_diffe
38752	2018-01-07	Estonia	Sweden	1	1	Friendly	Abu Dhabi	United Arab Emirates	Tie	0
38755	2018-01-11	Indonesia	Iceland	0	6	Friendly	Sleman	Indonesia	Iceland	6
38756	2018-01-14	Indonesia	Iceland	1	4	Friendly	Jakarta	Indonesia	Iceland	3
38757	2018-01-27	Moldova	Korea Republic	0	1	Friendly	Antalya	Turkey	Korea Republic	1
38759	2018-01-30	Jamaica	Korea Republic	2	2	Friendly	Antalya	Turkey	Tie	0
38762	2018-02-03	Latvia	Korea Republic	0	1	Friendly	Antalya	Turkey	Korea Republic	1
38764	2018-02-28	Iraq	Saudi Arabia	4	1	Friendly	Basra	Iraq	Iraq	3
38791	2018-03-23	Netherlands	England	0	1	Friendly	Amsterdam	Netherlands	England	1
38794	2018-03-23	Italy	Argentina	0	2	Friendly	Manchester	England	Argentina	2
38795	2018-03-23	Norway	Australia	4	1	Friendly	Oslo	Norway	Norway	3
38796	2018-03-23	Greece	Switzerland	0	1	Friendly	Athens	Greece	Switzerland	1
38803	2018-03-23	Ukraine	Saudi Arabia	1	1	Friendly	Marbella	Spain	Tie	0
38809	2018-03-23	Scotland	Costa Rica	0	1	Friendly	Glasgow	Scotland	Costa Rica	1
38825	2018-03-24	Northern Ireland	Korea Republic	2	1	Friendly	Belfast	Northern Ireland	Northern Ireland	1

	date	home_team	away_team	home_score	away_score	tournament	city	country	winning_team	goal_diffe
38851	2018-03-26	Wales	Uruguay	0	1	Friendly	Nanning	China	Uruguay	1
38867	2018-03-27	Greece	Egypt	1	0	Friendly	Zurich	Switzerland	Greece	1
38877	2018-03-27	Ukraine	Japan	2	1	Friendly	Liège	Belgium	Ukraine	1
38880	2018-03-27	Romania	Sweden	1	0	Friendly	Craiova	Romania	Romania	1
38881	2018-03-27	Bosnia-Herzegovina	Senegal	0	0	Friendly	Le Havre	France	Tie	0
38907	2018-06-01	Iran	Spain	0	1	Worldcup	Russia	Russia	Spain	1

16749 rows × 10 columns

As mentioned earlier, slicing our data from 1930 onwards, due to limitation of elo ranking dataset

In [6]:
```
# Loop for creating a new column 'year'
year = []
for row in df_teams['date']:
    year.append(int(row[:4]))
df_teams['match_year'] = year

# Slicing the dataset to see how many matches took place from 1930 onwards (the year of the first ever World Cup)
df_teams30 = df_teams[df_teams.match_year >= 1930]
df_teams30.head()
```

Out[6]:

	date	home_team	away_team	home_score	away_score	tournament	city	country	winning_team	goal_difference
1230	1930-01-01	Spain	Czechoslovakia	1	0	Friendly	Barcelona	Spain	Spain	1
1231	1930-01-12	Portugal	Czechoslovakia	1	0	Friendly	Lisbon	Portugal	Portugal	1
1237	1930-02-23	Portugal	France	2	0	Friendly	Porto	Portugal	Portugal	2
1238	1930-03-02	Germany	Italy	0	2	Friendly	Frankfurt am Main	Germany	Italy	2
1240	1930-03-23	France	Switzerland	3	3	Friendly	Colombes	France	Tie	0

Dropping unused column to reduce dimension needed for training (Speed up training)

In [7]:
```
df_teams30 = df_teams30.drop(['date', 'home_score', 'away_score', 'tournament', 'city', 'country', 'goal_difference',
'match_year'], axis=1)
df_teams30.head(5)
```

Out[7]:

	home_team	away_team	winning_team
1230	Spain	Czechoslovakia	Spain
1231	Portugal	Czechoslovakia	Portugal
1237	Portugal	France	Portugal
1238	Germany	Italy	Italy
1240	France	Switzerland	Tie

Map ELO rating based on team name, a major mistake made during the collection of ELO rating
(Only qualifier data was collected, in order to have a balance historical view even for teams that did not make it to the qualifiers we only took 2018 rating for missing countries)

In [8]:
```
# Read .csv files from elo rating
elorating = pd.read_csv('datasets/EloRating.csv', encoding = 'ISO-8859-1')

ELODict={}
for index, row in elorating.iterrows():
    ELODict[row["Team"]]= row["Rating"]

# map rating information into our dataframe
df_teams30['home_team_rating']=df_teams30['home_team'].map(ELODict)
df_teams30['away_team_rating']=df_teams30['away_team'].map(ELODict)
df_teams30['rating_diff'] = df_teams30['home_team_rating'] - df_teams30['away_team_rating']
df_teams30['rating_diff'] = df_teams30['rating_diff'].astype('int')

df_teams30.head()
```

Out[8]:

	home_team	away_team	winning_team	home_team_rating	away_team_rating	rating_diff
1230	Spain	Czechoslovakia	Spain	2038	1882	156
1231	Portugal	Czechoslovakia	Portugal	1976	1882	94
1237	Portugal	France	Portugal	1976	1999	-23
1238	Germany	Italy	Italy	2077	1850	227
1240	France	Switzerland	Tie	1999	1890	109

2) Building our ML Model

Before building the model, we split the data into x,y (X variable like home vs away pair and Y variable who wins)
probably should convert Y into 1 hot vector to avoid bias
Also swap out X variable of country name into ID via hashmap (Keras Input limitation)

In [9]:
```
# rename winning team string as integer
df_teams30 = df_teams30.reset_index(drop=True)
df_teams30.loc[df_teams30.winning_team == df_teams30.home_team, 'winning_team']= 0
df_teams30.loc[df_teams30.winning_team == 'Tie', 'winning_team']= 2
df_teams30.loc[df_teams30.winning_team == df_teams30.away_team, 'winning_team']= 1
df_teams30.head()
```

Out[9]:

	home_team	away_team	winning_team	home_team_rating	away_team_rating	rating_diff
0	Spain	Czechoslovakia	0	2038	1882	156
1	Portugal	Czechoslovakia	0	1976	1882	94
2	Portugal	France	0	1976	1999	-23
3	Germany	Italy	1	2077	1850	227
4	France	Switzerland	2	1999	1890	109

In [10]:
```
# convert all teams into unique ID via dictionary
finalX = df_teams30['home_team'].append(df_teams30['away_team'])
CountryDict = dict([(y,x+1) for x,y in enumerate(sorted(set(finalX)))])
df_teams30["home_team"].replace(CountryDict, inplace=True)
df_teams30["away_team"].replace(CountryDict, inplace=True)

# Separate X and y sets
X = df_teams30.drop(['winning_team'], axis=1)
y = df_teams30["winning_team"]
y = y.astype('int')

# Separate train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42, shuffle=True)

# 1 hotvectorisation of our Y column to avoid bias (Also for model to predict category)
y_test_1hot = to_categorical(y_test).astype(np.int)
y_train_1hot = to_categorical(y_train).astype(np.int)
```

basic logistic regression

In [11]:
```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
logreg.fit(X_train, y_train)
score = logreg.score(X_train, y_train)
score2 = logreg.score(X_test, y_test)

print("Training set accuracy: ", '%.3f'%(score))
print("Test set accuracy: ", '%.3f'%(score2))
```

```
Training set accuracy:  0.536
Test set accuracy:  0.542
```

Noticed model is only 50+% accuracy which is pretty low, slightly better than guessing even after using elo ranking dataset as briefly mentioned above to improve our model accuracy instead of just country ID

Alternatlve model in keras

Current stack consist input layer with 5 dimension (with elorating for away and home team)
Next dense fully connected dense layers with built in activation relu
Finally softmax to squash output prediction as probability between representing win, draw or tie

In [29]:
```
### define and fit the final model
model = Sequential()
model.add(Dense(32, input_dim=5, activation='relu'))
# add noise to model to avoid bais
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu',kernel_initializer='normal'))
model.add(Dense(128, activation='relu',kernel_initializer='normal'))
model.add(Dense(64, activation='relu',kernel_initializer='normal'))
model.add(Dense(32, activation='relu',kernel_initializer='normal'))
#model.add(Dense(3))
# squash result as probability
model.add(Dense(3, activation='softmax'))


#optimisation to converge faster
#epsilon so that division is not 0 think of it as bias recommended 10^-8
#rmsprop = RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0)
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['categorical_accuracy'])



# toggle verbose to print text
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=20, verbose=0,batch_size=32)
#model.fit(X_train, y_train_1hot, validation_data=(X_test,y_test_1hot), epochs=1000, verbose=0)
```

Out[29]:
```
<keras.callbacks.History at 0x1ddfdfa29b0>
```

In [30]:
```
# backup our model
model.save('model/my_model_current.h5')

# show summary of model
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_7 (Dense)              (None, 32)                192       
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0         
_________________________________________________________________
dense_8 (Dense)              (None, 64)                2112      
_________________________________________________________________
dense_9 (Dense)              (None, 128)               8320      
_________________________________________________________________
dense_10 (Dense)             (None, 64)                8256      
_________________________________________________________________
dense_11 (Dense)             (None, 32)                2080      
_________________________________________________________________
dense_12 (Dense)             (None, 3)                 99        
=================================================================
Total params: 21,059
Trainable params: 21,059
Non-trainable params: 0
_________________________________________________________________
```

Noticed we only have 50+ % accuracy not very good for a ML model which should at least hit 70. We will be including elo ranking dataset as briefly mentioned above to improve our model accuracy

In [31]:
```
scores = model.evaluate(X_train, y_train, verbose=0)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))

#print(y_test)
# sampling our model prediction
test_sample=np.array([145,183,1902,1795,178]).reshape(1,5)
# predict output index
sample_output=model.predict_classes(test_sample)
print(sample_output)

#import numpy
#from numpy import unravel_index
#numpy.set_printoptions(threshold=numpy.nan)
# result verification

#y_pred_probability = model.predict_proba(X_test)
#y_pred = model.predict_classes(X_test)

#print(y_pred)

#predictions = numpy.argmax(model.predict(X_test), axis=1)
#for i in range(1000):
#    print(predictions[i])
#plt.scatter(X_test['home_team'], y_pred)
#plt.show()
```

```
categorical_accuracy: 77.12%
[0]
```

3) Creating prediction sets from current 2018 data

before final prediction we will have to clean up the dataset and merge accordingly

In [32]:
```
# Loading new datasets
ranking = pd.read_csv('datasets/fifa_rankings.csv') # Obtained from https://us.soccerway.com/teams/rankings/fifa/?ICID
=TN_03_05_01
fixtures = pd.read_csv('datasets/fixtures.csv') # Obtained from https://fixturedownload.com/results/fifa-world-cup-201
8

# List for storing the group stage games
pred_set = []
```

include fix ranking within our 2018 group stage fixture further sort by ranking

In [33]:
```
# Create new columns with ranking position of each team
fixtures.insert(1, 'first_position', fixtures['Home Team'].map(ranking.set_index('Team')['Position']))
fixtures.insert(2, 'second_position', fixtures['Away Team'].map(ranking.set_index('Team')['Position']))

# We only need the group stage games, so we have to slice the dataset
# the slice can be read as get till row 48 for all columns
fixtures = fixtures.iloc[:48, :]
fixtures.tail()
```

Out[33]:

	Round Number	first_position	second_position	Date	Location	Home Team	Away Team	Group	Result
43	3	6.0	25.0	27/06/2018 21:00	Nizhny Novgorod Stadium	Switzerland	Costa Rica	Group E	NaN
44	3	60.0	10.0	28/06/2018 17:00	Volgograd Stadium	Japan	Poland	Group H	NaN
45	3	28.0	16.0	28/06/2018 17:00	Samara Stadium	Senegal	Colombia	Group H	NaN
46	3	55.0	14.0	28/06/2018 21:00	Saransk Stadium	Panama	Tunisia	Group G	NaN
47	3	13.0	3.0	28/06/2018 21:00	Kaliningrad Stadium	England	Belgium	Group G	NaN

predicting which team proceeds to next stage

In [34]:
```
# Loop to add teams to new prediction dataset based on the ranking position of each team# Loop
for index, row in fixtures.iterrows():
    if row['first_position'] < row['second_position']:
        pred_set.append({'home_team': row['Home Team'], 'away_team': row['Away Team'], 'winning_team': None})
    else:
        pred_set.append({'home_team': row['Away Team'], 'away_team': row['Home Team'], 'winning_team': None})

pred_set = pd.DataFrame(pred_set)
backup_pred_set = pred_set

pred_set.head()
```

Out[34]:

	away_team	home_team	winning_team
0	Saudi Arabia	Russia	None
1	Egypt	Uruguay	None
2	Morocco	Iran	None
3	Spain	Portugal	None
4	Australia	France	None

4) Deploy Model

Prepare match pairs from fixtures dataset (feed in eloranking from our dictionary)
Using our previously trained model to predict outcome

In [35]:
```
#load our previously trained model
model = load_model('model/my_model_current.h5')

# convert our group stage data into tuples
groupstage=pred_set.drop(['winning_team'], axis=1)
groupstagetuples = [tuple(x) for x in groupstage.values]


def prepare_predict(matches):

    wc_x_pred = []
    # data preprocessing
    for matchPairs in matches:
        home_team_id = CountryDict[matchPairs[0]]
        away_team_id = CountryDict[matchPairs[1]]
        home_team_elorating = ELODict[matchPairs[0]]
        away_team_elorating = ELODict[matchPairs[1]]
        elodifference = ELODict[matchPairs[0]]-ELODict[matchPairs[1]]

        # transform array
        matchset = [home_team_id, away_team_id, home_team_elorating, away_team_elorating, elodifference]
        wc_x_pred.append(matchset)

    # convert prediction set into numpy array
    wc_x_pred = np.array(wc_x_pred)

    # return y_prediction probability
    wc_y_pred = model.predict_proba(wc_x_pred)
    # iterate results
    for index, outcome in enumerate(wc_y_pred):
        outcome_win = str(outcome[0])
        outcome_draw = str(outcome[2])
        outcome_lose = str(outcome[1])
        outcome_final = np.where(outcome == outcome.max())


        print('Probability of ' + matches[index][0]+ ' winning: ' + outcome_win)
        print('Probability of Tie: '+ outcome_draw)
        print('Probability of ' + matches[index][1] + ' winning: ' + outcome_lose)
        if(outcome_final[0]==0):
            print("Final Winner: "+ matches[index][0])
```

```
        elif(outcome_final[0]==1):
            print("Final Winner: "+ matches[index][1])
        else:
            print("Draw")
        print("\n")


prepare_predict(groupstagetuples)
```

```
Probability of Saudi Arabia winning: 0.38029093
Probability of Tie: 0.28141072
Probability of Russia winning: 0.3382984
Final Winner: Saudi Arabia


Probability of Egypt winning: 0.30319843
Probability of Tie: 0.26603892
Probability of Uruguay winning: 0.43076265
Final Winner: Uruguay


Probability of Morocco winning: 0.39537862
Probability of Tie: 0.28471026
Probability of Iran winning: 0.31991115
Final Winner: Morocco


Probability of Spain winning: 0.49481034
Probability of Tie: 0.26638287
Probability of Portugal winning: 0.23880678
Final Winner: Spain


Probability of Australia winning: 0.2968698
Probability of Tie: 0.2641031
Probability of France winning: 0.43902707
Final Winner: France


Probability of Iceland winning: 0.33922574
Probability of Tie: 0.27757338
Probability of Argentina winning: 0.38320082
Final Winner: Argentina


Probability of Denmark winning: 0.4826675
Probability of Tie: 0.2700492
Probability of Peru winning: 0.2472832
Final Winner: Denmark
```

```
Probability of Nigeria winning: 0.33186263
Probability of Tie: 0.27548844
Probability of Croatia winning: 0.39264897
Final Winner: Croatia


Probability of Serbia winning: 0.49680924
Probability of Tie: 0.2657671
Probability of Costa Rica winning: 0.2374236
Final Winner: Serbia


Probability of Mexico winning: 0.3302847
Probability of Tie: 0.2749838
Probability of Germany winning: 0.3947315
Final Winner: Germany


Probability of Switzerland winning: 0.32265916
Probability of Tie: 0.2728392
Probability of Brazil winning: 0.40450165
Final Winner: Brazil


Probability of Korea Republic winning: 0.41112116
Probability of Tie: 0.2869204
Probability of Sweden winning: 0.3019585
Final Winner: Korea Republic


Probability of Panama winning: 0.2965419
Probability of Tie: 0.2642543
Probability of Belgium winning: 0.4392039
Final Winner: Belgium


Probability of Tunisia winning: 0.30554542
Probability of Tie: 0.26727253
Probability of England winning: 0.42718205
Final Winner: England
```

```
Probability of Japan winning: 0.3178484
Probability of Tie: 0.27126318
Probability of Colombia winning: 0.4108884
Final Winner: Colombia


Probability of Senegal winning: 0.41134262
Probability of Tie: 0.28795582
Probability of Poland winning: 0.3007015
Final Winner: Senegal


Probability of Russia winning: 0.49064633
Probability of Tie: 0.26765457
Probability of Egypt winning: 0.24169913
Final Winner: Russia


Probability of Morocco winning: 0.315025
Probability of Tie: 0.27014878
Probability of Portugal winning: 0.41482624
Final Winner: Portugal


Probability of Saudi Arabia winning: 0.28064996
Probability of Tie: 0.2578574
Probability of Uruguay winning: 0.46149263
Final Winner: Uruguay


Probability of Iran winning: 0.3332706
Probability of Tie: 0.27572933
Probability of Spain winning: 0.391
Final Winner: Spain


Probability of Australia winning: 0.3519812
Probability of Tie: 0.2792029
Probability of Denmark winning: 0.36881587
Final Winner: Denmark
```

```
Probability of Peru winning: 0.39126843
Probability of Tie: 0.28286007
Probability of France winning: 0.32587156
Final Winner: Peru


Probability of Croatia winning: 0.40353304
Probability of Tie: 0.28728724
Probability of Argentina winning: 0.30917975
Final Winner: Croatia


Probability of Costa Rica winning: 0.20057718
Probability of Tie: 0.22020537
Probability of Brazil winning: 0.57921743
Final Winner: Brazil


Probability of Nigeria winning: 0.38465372
Probability of Tie: 0.28225392
Probability of Iceland winning: 0.33309236
Final Winner: Nigeria


Probability of Serbia winning: 0.41100517
Probability of Tie: 0.28780243
Probability of Switzerland winning: 0.3011924
Final Winner: Serbia


Probability of Tunisia winning: 0.29930907
Probability of Tie: 0.26528254
Probability of Belgium winning: 0.4354084
Final Winner: Belgium


Probability of Korea Republic winning: 0.37672982
Probability of Tie: 0.28209516
Probability of Mexico winning: 0.34117505
Final Winner: Korea Republic
```

```
Probability of Sweden winning: 0.31354585
Probability of Tie: 0.2699044
Probability of Germany winning: 0.41654974
Final Winner: Germany


Probability of Panama winning: 0.30019346
Probability of Tie: 0.26538742
Probability of England winning: 0.43441907
Final Winner: England


Probability of Japan winning: 0.415339
Probability of Tie: 0.2869112
Probability of Senegal winning: 0.29774985
Final Winner: Japan


Probability of Colombia winning: 0.52545655
Probability of Tie: 0.25657114
Probability of Poland winning: 0.21797231
Final Winner: Colombia


Probability of Russia winning: 0.3577389
Probability of Tie: 0.27952498
Probability of Uruguay winning: 0.36273614
Final Winner: Uruguay


Probability of Saudi Arabia winning: 0.40731514
Probability of Tie: 0.2872009
Probability of Egypt winning: 0.3054839
Final Winner: Saudi Arabia


Probability of Iran winning: 0.36247486
Probability of Tie: 0.28042138
Probability of Portugal winning: 0.3571037
Final Winner: Iran
```

```
Probability of Morocco winning: 0.28632322
Probability of Tie: 0.2600842
Probability of Spain winning: 0.45359254
Final Winner: Spain


Probability of Denmark winning: 0.39557478
Probability of Tie: 0.28649512
Probability of France winning: 0.31793004
Final Winner: Denmark


Probability of Australia winning: 0.36941645
Probability of Tie: 0.2812454
Probability of Peru winning: 0.3493381
Final Winner: Australia


Probability of Nigeria winning: 0.27299628
Probability of Tie: 0.2553461
Probability of Argentina winning: 0.47165772
Final Winner: Argentina


Probability of Iceland winning: 0.3999347
Probability of Tie: 0.28708062
Probability of Croatia winning: 0.31298465
Final Winner: Iceland


Probability of Sweden winning: 0.4142569
Probability of Tie: 0.287553
Probability of Mexico winning: 0.29819015
Final Winner: Sweden


Probability of Korea Republic winning: 0.25341946
Probability of Tie: 0.24671608
Probability of Germany winning: 0.49986443
Final Winner: Germany
```

```
Probability of Serbia winning: 0.26934028
Probability of Tie: 0.2538205
Probability of Brazil winning: 0.4768392
Final Winner: Brazil


Probability of Costa Rica winning: 0.3529764
Probability of Tie: 0.2790482
Probability of Switzerland winning: 0.36797538
Final Winner: Switzerland


Probability of Japan winning: 0.36970147
Probability of Tie: 0.28132504
Probability of Poland winning: 0.3489735
Final Winner: Japan


Probability of Senegal winning: 0.35286307
Probability of Tie: 0.27952954
Probability of Colombia winning: 0.36760733
Final Winner: Colombia


Probability of Panama winning: 0.45323938
Probability of Tie: 0.27816173
Probability of Tunisia winning: 0.26859885
Final Winner: Panama


Probability of England winning: 0.47497857
Probability of Tie: 0.2721127
Probability of Belgium winning: 0.2529088
Final Winner: England
```

hardcoded the group stage tuple all the way to the finals, this sort of flexibility instead of code driven function allows us to modify who proceed to quater finals so on and so forth based on actual results (Ideally, our model should predict the outcode for every match correctly but hey nothing is perfect right?)

Note: Replaced some of the wrongly predicted outcomes argentina with respective country

Group knockoff:

1A vs 2B

1C vs 2D

1E vs 2F

1G vs 2H

1B vs 2A

1D vs 2C

1F vs 2E

1H vs 2G

In [36]:
```
# List of tuples before we arrange the teams in home and away
group_16 = [('Uruguay', 'Portugal'),
            ('France', 'Argentina'),
            ('Brazil', 'Germany'),
            ('Belgium', 'Senegal'),
            ('Spain', 'Russia'),
            ('Croatia', 'Denmark'),
            ('Mexico', 'Switzerland'),
            ('Japan', 'England')]
```

Function to clean tuple dataset from fixture and order by ranking (A not so nice approach to give higher ranking teams as homedue to higher win rate)

In [37]:
```
# actual run
prepare_predict(group_16)
```

```
Probability of Uruguay winning: 0.41393515
Probability of Tie: 0.287764
Probability of Portugal winning: 0.29830077
Final Winner: Uruguay


Probability of France winning: 0.48632455
Probability of Tie: 0.26895857
Probability of Argentina winning: 0.24471697
Final Winner: France


Probability of Brazil winning: 0.51679933
Probability of Tie: 0.25942263
Probability of Germany winning: 0.22377811
Final Winner: Brazil


Probability of Belgium winning: 0.57454264
Probability of Tie: 0.23925507
Probability of Senegal winning: 0.18620227
Final Winner: Belgium


Probability of Spain winning: 0.64453065
Probability of Tie: 0.21130899
Probability of Russia winning: 0.14416035
Final Winner: Spain


Probability of Croatia winning: 0.46185663
Probability of Tie: 0.27572486
Probability of Denmark winning: 0.2624185
Final Winner: Croatia


Probability of Mexico winning: 0.42954183
Probability of Tie: 0.28440252
Probability of Switzerland winning: 0.28605568
Final Winner: Mexico
```

```
Probability of Japan winning: 0.30955642
Probability of Tie: 0.268546
Probability of England winning: 0.42189756
Final Winner: England
```

based on previous result proceed

In [38]:
```
# List of matches
quarters = [('Uruguay', 'France'),
            ('Brazil', 'Belgium'),
            ('Spain', 'Croatia'),
            ('Mexico', 'England')]
```

In [39]:
```
prepare_predict(quarters)
```

```
Probability of Uruguay winning: 0.40576446
Probability of Tie: 0.28583232
Probability of France winning: 0.30840322
Final Winner: Uruguay


Probability of Brazil winning: 0.5795593
Probability of Tie: 0.23737769
Probability of Belgium winning: 0.18306299
Final Winner: Brazil


Probability of Spain winning: 0.54230297
Probability of Tie: 0.25084627
Probability of Croatia winning: 0.20685077
Final Winner: Spain


Probability of Mexico winning: 0.39773363
Probability of Tie: 0.28659722
Probability of England winning: 0.31566918
Final Winner: Mexico
```

In [40]:
```
# List of matches
semi = [('Uruguay', 'Brazil'),
        ('Spain', 'Mexico')]
```

In [41]:
```
prepare_predict(semi)
```

```
Probability of Uruguay winning: 0.3302968
Probability of Tie: 0.27512553
Probability of Brazil winning: 0.39457765
Final Winner: Brazil


Probability of Spain winning: 0.5632736
Probability of Tie: 0.2434003
Probability of Mexico winning: 0.19332607
Final Winner: Spain
```

In [42]:
```
# List of matches
losersfinals = [('Uruguay', 'Mexico')]
```

In [43]:
```
prepare_predict(losersfinals)
```

```
Probability of Uruguay winning: 0.48236373
Probability of Tie: 0.27013937
Probability of Mexico winning: 0.24749702
Final Winner: Uruguay
```

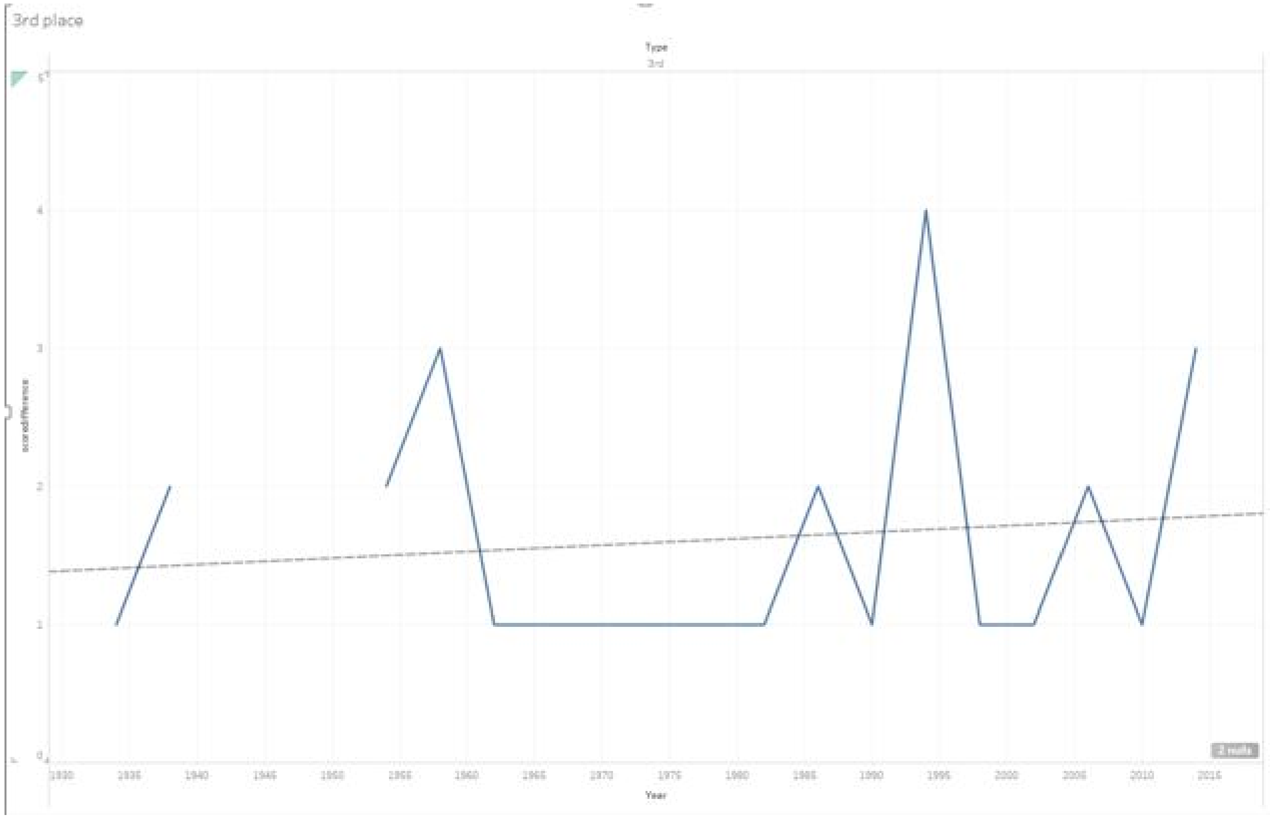
In [44]:
```
# The final game# The big
finals = [('Brazil', 'Spain')]
```

In [45]:
```
prepare_predict(finals)
```

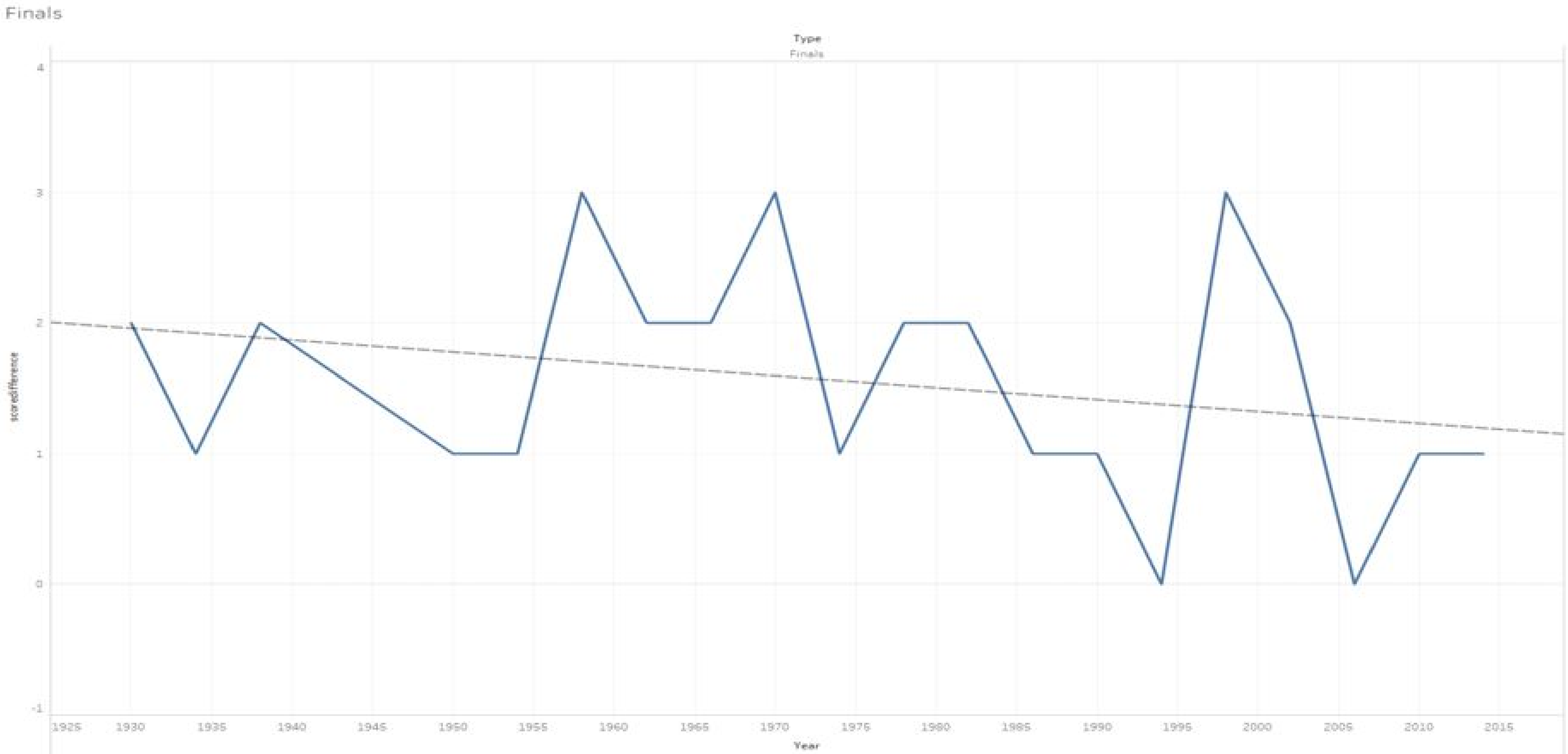
```
Probability of Brazil winning: 0.53687376
Probability of Tie: 0.25271645
Probability of Spain winning: 0.21040975
Final Winner: Brazil
```

5) FIFA 2018 Score Prediction

Using simple linear regression to predict the Total Goals for the Final/SemiFinal Match
After which a difference of winner loser linear regression is used based on the predicted Total Goals to retrieve the end result
For more detail please refer to: 2018 world cup score prediction/(Total)



The predicted Score for the 3rd place playoff: 4-3



The predicted Score for the Finals: 2-1

6) Project Review

The experiments:
Elo rating did improve prediction from 50~% to 70~%
found that LSTM model would not help with model converging
the score prediction was separated out from winners and losers to avoid dependancies

For future work:
Improve handling of draw to proceed in matches after group stage
Also ranking could be used for training and not just sorting teams between home and away
Finally, betting historic data would have been helpful

Limitation
The limitation of this model would be the inability to predict score but instead predicting win or lose
A separate linear regression algorithm would be use to predict the score instead

Results
1st Brazil, 2nd Spain, 3rd Uruguay
Final Match: 2-1
Losers Final: 4-3

7) References

https://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/ (https://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/)
https://medium.com/@siavash_37715/how-to-predict-bitcoin-and-ethereum-price-with-rnn-lstm-in-keras-a6d8ee8a5109 (https://medium.com/@siavash_37715/how-to-predict-bitcoin-and-ethereum-price-with-rnn-lstm-in-keras-a6d8ee8a5109)
https://towardsdatascience.com/using-lstms-to-forecast-time-series-4ab688386b1f (https://towardsdatascience.com/using-lstms-to-forecast-time-series-4ab688386b1f)
https://github.com/charliechurches/Russia_World_Cup_Prediction/blob/master/Russia_World_Cup_predict.ipynb (https://github.com/charliechurches/Russia_World_Cup_Prediction/blob/master/Russia_World_Cup_predict.ipynb)
https://www.eloratings.net/ (https://www.eloratings.net/)
https://www.youtube.com/watch?v=Cltt47Ah3Q4 (https://www.youtube.com/watch?v=Cltt47Ah3Q4)