In [1]:
```
import pandas as pd
import numpy as np
import math
import matplotlib
import matplotlib.pyplot as plt
import sklearn.metrics as metrics
from sklearn import ensemble
from sklearn.model_selection import cross_val_score
from sklearn import tree
from copy import deepcopy
from sklearn.model_selection import KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from bs4 import BeautifulSoup
import requests
import time
```

In [2]:
```
#function to get all game ids for one team
def team_schedule(team_name, months):
    game_ids = []
    for m in months:
        url1 = "https://www.basketball-reference.com/leagues/NBA_2018_games-"
        req = requests.get(url1 + m + '.html')
        page = req.text
        soup = BeautifulSoup(page, 'html.parser')
        rows = soup.find_all('tr')
        for row in rows:
            team_cells = row.find_all('td', 'left')
            for cell in team_cells:
                if team_name in cell.get_text():
                    center_cell = row.find('td', 'center')
                    if center_cell.get_text() == "Box Score":
                        link = center_cell.find_all('a')[0]
                        url = link.get('href')
                        game_id = url.split('/')[2]
                        game_ids.append(game_id)
    return game_ids

def to_pbp_url(game_id):
    return "https://www.basketball-reference.com/boxscores/pbp/" + game_id
```

In [3]:
```
months = ['october', 'november']
# schedule_BOS = team_schedule('Boston Celtics', months)
# BOS_pbp_urls = list(map(to_pbp_url, schedule_BOS))
```

In [4]:
```
teams = ['Boston Celtics', 'Cleveland Cavaliers', 'Toronto Raptors']
# 'Brooklyn Nets', 'Golden State Warriors', 'Miami Heat', 'Houston Rockets', 'Chi
schedules = []
for team in teams:
    schedules.append(team_schedule(team, months))
    time.sleep(5)
pbp_urls = []
for schedule in schedules:
    pbp_urls = (list(map(to_pbp_url, schedule))) + pbp_urls
```

In [5]:
```
#delete all duplicate games
pbp_urls = list(set(pbp_urls))
```

In [6]:
```
dataframes = []
for url in pbp_urls:
    r = requests.get(url)
    if r.status_code != 404:
        pbp = pd.read_html(url,header=1)[0]
        #pbp.columns = ['time', 'awayevents','awaypts','score','homepts','homeeve
        dataframes.append(pbp)
    time.sleep(5)

dataframes[0].head()
```

Out[6]:

	Time	Orlando	Unnamed: 2	Score	Unnamed: 4	Cleveland
0	12:00.0	Start of 1st quarter	NaN	NaN	NaN	NaN
1	12:00.0	Jump ball: K. Love vs. N. Vucevic (E. Fournier...	NaN	NaN	NaN	NaN
2	11:49.0	N. Vucevic makes 2-pt shot at rim (assist by T...	2.0	2-0	NaN	NaN
3	11:33.0	NaN	NaN	2-0	NaN	K. Love misses 2-pt shot from 17 ft
4	11:30.0	Defensive rebound by N. Vucevic	NaN	2-0	NaN	NaN

In [7]:
```
def time_to_int(t):
    if len(t) > 5:
        numbers = t.split(":")
        seconds = numbers[1].split(".")
        t_sec = (int(numbers[0])*60) + int(seconds[0])
    else:
        t_sec = None
    return t_sec
```

In [8]:
```
#function to format scraped data
def format_frames(pbp):
    awayteam = pbp.columns[1]
    hometeam = pbp.columns[5]
    pbp.columns = ['time', 'awayevents','awaypts','score','homepts','homeevents']
    pbp['awayteam'] = awayteam
    pbp['hometeam'] = hometeam
    events = pbp['awayevents']
    events = events.fillna(pbp['homeevents'])
    pbp['event'] = events
    #possession with 1 indicating that the away team is making the play and 0 ina
    pbp['possession'] = 1-pd.isnull(pbp['awayevents'])
    #first need to replace scores at the beginning of games and the rows that jus
    pbp['score'] = pbp['score'].replace(to_replace='Score',method='ffill')
    pbp['score'] = pbp['score'].fillna(method='bfill')

    #then we split these to the away team scores and home team scores
    #awayscore,homescore = score
    #print([len(x.split('-')) for x in pbp['score']])
    score = [x.split('-') for x in pbp['score']]
    awayscore,homescore = np.transpose(np.array(score))
    awayscore = [int(x) for x in awayscore]
    homescore = [int(x) for x in homescore]
    pbp['awayscore'] = awayscore
    pbp['homescore'] = homescore

    #now drop the redundant variables
    pbp = pbp.drop(['awayevents','awaypts','score','homepts','homeevents'], axis=
    pbp1 = pbp[pd.notnull(pbp['event'])]
    #adding quarter to dataframe
    quarter = []
    count2 = 1
    for x, y in zip(pbp1['event'], pbp1['time']):
        if 'End of' in x and '0:00.0' in y:
            count2 += 1;
        quarter.append(count2)

    pbp1['quarter'] = quarter
    #creating Time Outs Remaining columns and point streak column
    for i, x in enumerate(['home', 'away']):
        if i == 1:
            team = awayteam
        else:
            team = hometeam
        TOR = []
        count = 6
        for y in pbp1['event']:
            if (team + " full timeout") in y:
                count = count - 1;
            TOR.append(count)

        pbp1['TOR_' + x] = TOR
        #point streak
        streak = []
        count1 = 0
        for i, (q, p) in enumerate(zip(pbp1["awayscore"], pbp1["homescore"])):
```

```
            if q > pbp1['awayscore'].iloc[i - 1] and p == pbp1['homescore'].iloc[
                count1 += 1
            elif p > pbp1['homescore'].iloc[i - 1] and q == pbp1['awayscore'].ilc
                count1 = 0
            streak.append(count1)

        pbp1[x + "_streak"] = streak
    pbp1['time'] = pbp1['time'].apply(time_to_int)
    #getting rid of time = null rows
    pbp2 = pbp1[pd.notnull(pbp1['time'])]
    total_time = []
    for x, y in zip(pbp2['quarter'], pbp2['time']):
        total_time.append((5-x)*y)
    pbp2["total_time"] = total_time
    away_final = np.max(pbp2['awayscore'])
    home_final = np.max(pbp2['homescore'])
    if away_final > home_final:
        pbp2['win'] = pbp2['possession']
    else:
        pbp2['win'] = [1 if x==0 else 0 for x in pbp2['possession']]
    return pbp2
```

In [9]:
```
formatted_frames = list(map(format_frames, dataframes))
```

```
C:\Users\charl\Anaconda3\lib\site-packages\ipykernel_launcher.py:38: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pa
ndas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\charl\Anaconda3\lib\site-packages\ipykernel_launcher.py:52: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pa
ndas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\charl\Anaconda3\lib\site-packages\ipykernel_launcher.py:63: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pa
ndas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\charl\Anaconda3\lib\site-packages\ipykernel_launcher.py:64: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pa
ndas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\charl\Anaconda3\lib\site-packages\ipykernel_launcher.py:70: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pa
ndas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\charl\Anaconda3\lib\site-packages\ipykernel_launcher.py:74: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pa
ndas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\charl\Anaconda3\lib\site-packages\ipykernel_launcher.py:76: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
```

stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)

In [10]:
```
for i, dataframe in enumerate(formatted_frames):
    dataframe["game_id"] = i
season_df = pd.concat(formatted_frames, ignore_index = True)
season_df.head()
```

Out[10]:

ime	awayteam	hometeam	event	possession	awayscore	homescore	quarter	TOR_home	hom
20.0	Orlando	Cleveland	Start of 1st quarter	1	2	0	1	6	
20.0	Orlando	Cleveland	Jump ball: K. Love vs. N. Vucevic (E. Fournier...	1	2	0	1	6	
09.0	Orlando	Cleveland	N. Vucevic makes 2-pt shot at rim (assist by T...	1	2	0	1	6	
93.0	Orlando	Cleveland	K. Love misses 2-pt shot from 17 ft	0	2	0	1	6	
90.0	Orlando	Cleveland	Defensive rebound by N. Vucevic	1	2	0	1	6	

In [11]:
```
#create a reference dataframe that holds time, event, game_id, and quarter and th
season_reference = season_df[['game_id', 'time', 'quarter', 'event', 'total_time'
season_reference.head()
```

Out[11]:

	game_id	time	quarter	event
0	0	720.0	1	Start of 1st quarter
1	0	720.0	1	Jump ball: K. Love vs. N. Vucevic (E. Fournier...
2	0	709.0	1	N. Vucevic makes 2-pt shot at rim (assist by T...
3	0	693.0	1	K. Love misses 2-pt shot from 17 ft
4	0	690.0	1	Defensive rebound by N. Vucevic

In [12]:
```
season_df = season_df.drop(['time', 'event', 'quarter'], axis=1)
season_df.head()
```

Out[12]:

	awayteam	hometeam	possession	awayscore	homescore	TOR_home	home_streak	TOR_away
0	Orlando	Cleveland	1	2	0	6	0	6
1	Orlando	Cleveland	1	2	0	6	0	6
2	Orlando	Cleveland	1	2	0	6	0	6
3	Orlando	Cleveland	0	2	0	6	0	6
4	Orlando	Cleveland	1	2	0	6	0	6

In [13]:
```
season_df.to_csv("season.csv", sep = ',')
season_reference.to_csv("season_reference.csv", sep = ',')
```

Random Forest

In [46]:
```
season_df = pd.read_csv('season.csv')
season_reference = pd.read_csv('season_reference.csv')
```

In [47]:
```
season_df.columns
```

Out[47]:
```
Index(['Unnamed: 0', 'awayteam', 'hometeam', 'possession', 'awayscore',
       'homescore', 'TOR_home', 'home_streak', 'TOR_away', 'away_streak',
       'total_time', 'win', 'game_id'],
      dtype='object')
```

In [48]:
```
season_df = season_df.drop('Unnamed: 0', axis = 1)
season_reference = season_reference.drop('Unnamed: 0', axis = 1)
test_games = season_df.loc[season_df['game_id'] >= 60]
```

In [49]:
```
msk = np.random.rand(len(season_df)) < 0.75
df_train = season_df[msk]
df_test = season_df[~msk]
```

In [50]:
```
#we have to one-hot encode every team, because decision trees implemented in scik
#and these features are interpreted always as continuous numeric variables

# one-hot encoding awayteam
one_hota = pd.get_dummies(df_train['awayteam'], prefix = 'away')
df_train1 = df_train.join(one_hota)
df_train1.columns
df_train1 = df_train1.drop('away_Toronto', axis = 1)

one_hota = pd.get_dummies(df_test['awayteam'], prefix = 'away')
df_test1 = df_test.join(one_hota)
df_test1 = df_test1.drop('away_Toronto', axis = 1)

#one-hot encoding hometeam
one_hoth = pd.get_dummies(df_train['hometeam'], prefix = "home")
df_train1 = df_train1.join(one_hoth)
df_train1 = df_train1.drop('home_Philadelphia', axis = 1)

one_hoth = pd.get_dummies(df_test['hometeam'], prefix = 'home')
df_test1 = df_test1.join(one_hoth)
df_test1 = df_test1.drop('home_Philadelphia', axis = 1)

df_train1 = df_train1.drop(['awayteam', 'hometeam'], axis = 1)
df_test1 = df_test1.drop(['awayteam', 'hometeam'], axis = 1)
```

In [51]:
```
# splitting into x and y training and testing
x_train = df_train1.loc[:, df_train1.columns != 'win']
y_train = df_train1['win']
x_test = df_test1.loc[:, df_test1.columns != 'win']
y_test = df_test1['win']
test_games_x = test_games1.loc[:, test_games1.columns != 'win']
```

In [53]:
```
#the square root of 53 is about 7, so we use 7 predictors for each tree
tree_counts2 = [2**x for x in range(1, 9)]
train_scores = dict()
test_scores = dict()
for c in tree_counts2:
    rand_forest = RandomForestClassifier(n_estimators = c, max_features = 7, ranc
    rand_forest.fit(x_train, y_train)
    tr_pred = rand_forest.predict(x_train)
    tst_pred = rand_forest.predict(x_test)
    train_scores[str(c)] = metrics.accuracy_score(y_train, tr_pred)
    test_scores[str(c)] = metrics.accuracy_score(y_test, tst_pred)
test_scores
```

Out[53]:
```
{'128': 1.0,
 '16': 0.99958199804932424,
 '2': 0.89772885606799502,
 '256': 1.0,
 '32': 1.0,
 '4': 0.97366587710742647,
 '64': 1.0,
 '8': 0.99540197854256651}
```

In [25]:
```
def get_rates(confusion_matrix):
    rates = dict()
    TN = confusion_matrix[0][0]
    FN = confusion_matrix[0][1]
    TP = confusion_matrix[1][1]
    FP = confusion_matrix[1][0]
    rates["True Positive Rate"] = (TP/(TP+FN))
    rates["False Positive Rate"] = (FP/(FP+TN))
    rates["True Negative Rate"] = (TN/(FP+TN))
    return rates
```

In [26]:
```
rand_forest = RandomForestClassifier(n_estimators = 8, max_features = 5, random_s
rand_forest.fit(x_train, y_train)
con_matrix = metrics.confusion_matrix(y_test, rand_forest.predict(x_test))
print(get_rates(con_matrix))
```

```
{'True Positive Rate': 0.99940101826894279, 'False Positive Rate': 0.0037967289
719626168, 'True Negative Rate': 0.99620327102803741}
```

In [130]:
```
test_games.loc[test_games['game_id'] == 61]
```

Out[130]:

	awayteam	hometeam	possession	awayscore	homescore	TOR_home	home_streak	TOR_
27677	New York	Cleveland	1	0	0	6	0	
27678	New York	Cleveland	1	0	0	6	0	
27679	New York	Cleveland	1	0	0	6	0	
27680	New York	Cleveland	0	0	2	6	0	
27681	New York	Cleveland	1	2	2	6	1	
27682	New York	Cleveland	0	2	2	6	1	
27683	New York	Cleveland	1	2	2	6	1	
27684	New York	Cleveland	1	4	2	6	2	
27685	New York	Cleveland	0	4	2	6	2	
27686	New York	Cleveland	1	4	2	6	2	
27687	New York	Cleveland	1	4	2	6	2	
27688	New York	Cleveland	0	4	2	6	2	
27689	New York	Cleveland	0	4	2	6	2	
27690	New York	Cleveland	1	4	2	6	2	
27691	New York	Cleveland	0	4	2	6	2	
27692	New York	Cleveland	1	4	2	6	2	
27693	New York	Cleveland	0	4	2	6	2	
27694	New York	Cleveland	0	4	2	6	2	
27695	New York	Cleveland	1	4	2	6	2	
27696	New York	Cleveland	0	4	2	6	2	
27697	New York	Cleveland	0	4	3	6	0	
27698	New York	Cleveland	0	4	4	6	0	
27699	New York	Cleveland	0	4	4	6	0	
27700	New York	Cleveland	1	6	4	6	1	
27701	New York	Cleveland	0	6	4	6	1	
27702	New York	Cleveland	1	6	4	6	1	
27703	New York	Cleveland	1	8	4	6	2	
27704	New York	Cleveland	0	8	7	6	0	
27705	New York	Cleveland	1	10	7	6	1	
27706	New York	Cleveland	0	10	10	6	0	
...	...	...	...	...	...	...	...	
28111	New York	Cleveland	0	106	93	2	1	
28112	New York	Cleveland	0	106	93	1	1	

	awayteam	hometeam	possession	awayscore	homescore	TOR_home	home_streak	TOR_
28113	New York	Cleveland	0	106	93	1	1	
28114	New York	Cleveland	0	106	93	1	1	
28115	New York	Cleveland	0	106	93	1	1	
28116	New York	Cleveland	1	108	93	1	2	
28117	New York	Cleveland	0	108	93	1	2	
28118	New York	Cleveland	1	108	93	1	2	
28119	New York	Cleveland	1	111	93	1	3	
28120	New York	Cleveland	0	111	93	1	3	
28121	New York	Cleveland	1	111	93	1	3	
28122	New York	Cleveland	0	111	93	1	3	
28123	New York	Cleveland	0	111	93	1	3	
28124	New York	Cleveland	0	111	93	1	3	
28125	New York	Cleveland	0	111	93	1	3	
28126	New York	Cleveland	1	111	93	1	3	
28127	New York	Cleveland	1	111	93	1	3	
28128	New York	Cleveland	1	111	93	1	3	
28129	New York	Cleveland	1	111	93	1	3	
28130	New York	Cleveland	1	111	93	1	3	
28131	New York	Cleveland	1	111	93	1	3	
28132	New York	Cleveland	0	111	93	1	3	
28133	New York	Cleveland	1	111	93	1	3	
28134	New York	Cleveland	1	114	93	1	4	
28135	New York	Cleveland	0	114	95	1	0	
28136	New York	Cleveland	1	114	95	1	0	
28137	New York	Cleveland	0	114	95	1	0	
28138	New York	Cleveland	0	114	95	1	0	
28139	New York	Cleveland	1	114	95	1	0	
28140	New York	Cleveland	1	114	95	1	0	

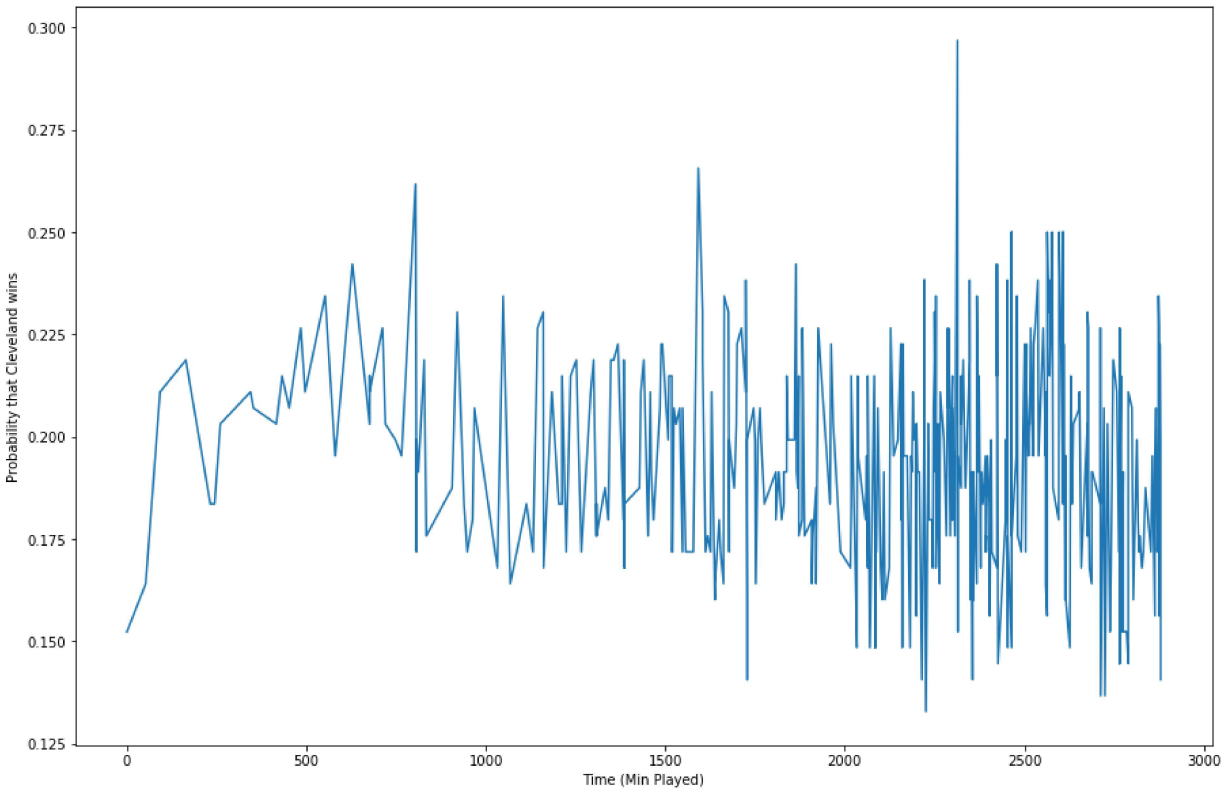
464 rows × 12 columns

In [79]:
```
cols = list(x_train.columns)
```

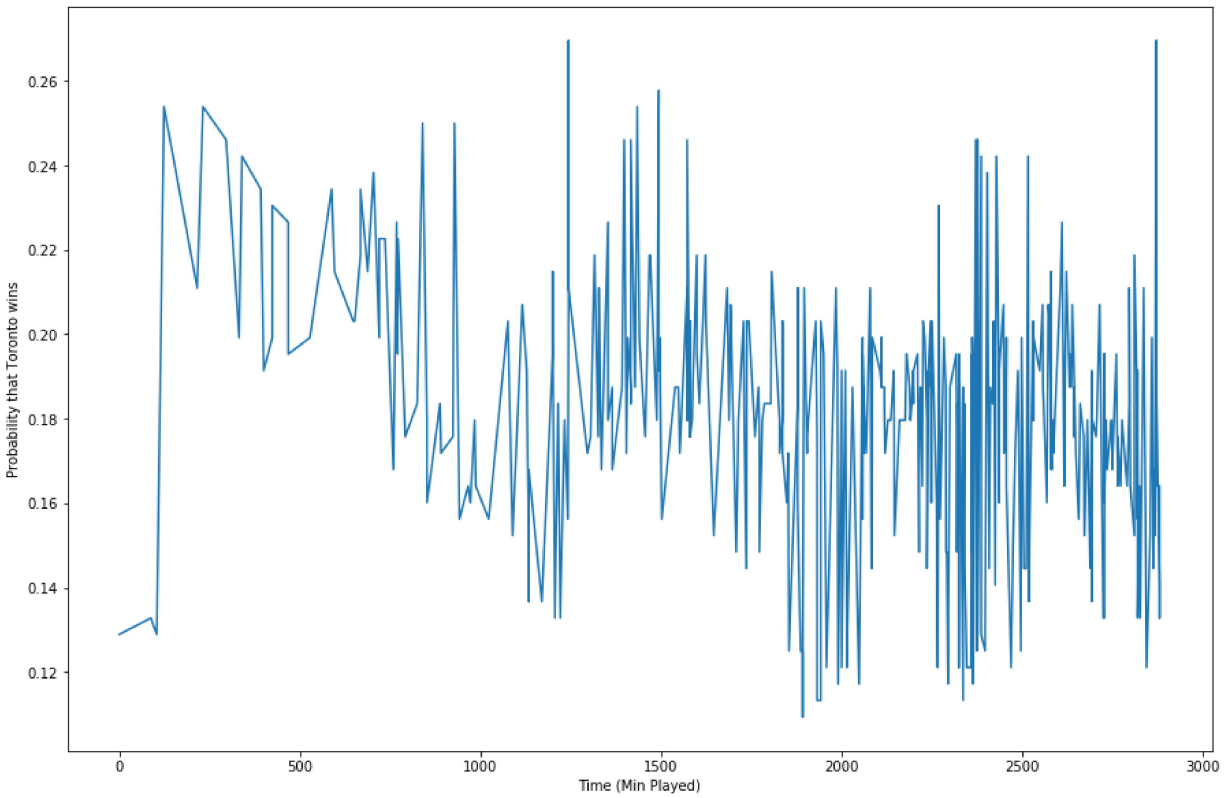
In [80]:
```
test_games_x = test_games.loc[:, test_games.columns != 'win']
```

In [125]:
```
def graph_game(game_id, city, model):
    game = test_games_x.loc[test_games['game_id'] == game_id]
    l_game = game.shape[0]
    one_hota = pd.get_dummies(game['awayteam'], prefix = 'away')
    game1 = game.join(one_hota)
    one_hoth = pd.get_dummies(game['hometeam'], prefix = "home")
    game1 = game1.join(one_hoth)
    game1 = game1.drop(['hometeam', 'awayteam',], axis = 1)
    game1 = game1.sort_values(by = 'total_time', ascending = True)
    game_cols = list(game1.columns)
    for col in cols:
        if col not in game_cols:
            game1[col] = np.zeros(l_game)
    wps = model.predict_proba(game1)
    poss = game1["possession"]
    if city in game['hometeam']:
        for i, (j, q) in enumerate(zip(wps, poss)):
            if q == 1:
                wps[i] = 1 - j
    else:
        for i, (j, q) in enumerate(zip(wps, poss)):
            if q == 0:
                wps[i] = 1 - j
    fig = plt.figure(figsize=(15, 10))
    plt.plot([2880 - x for x in game1['total_time']], [i[0] for i in wps])
    plt.xlabel('Time (Min Played)')
    plt.ylabel('Probability that '+ city + ' wins')
    plt.show()
```

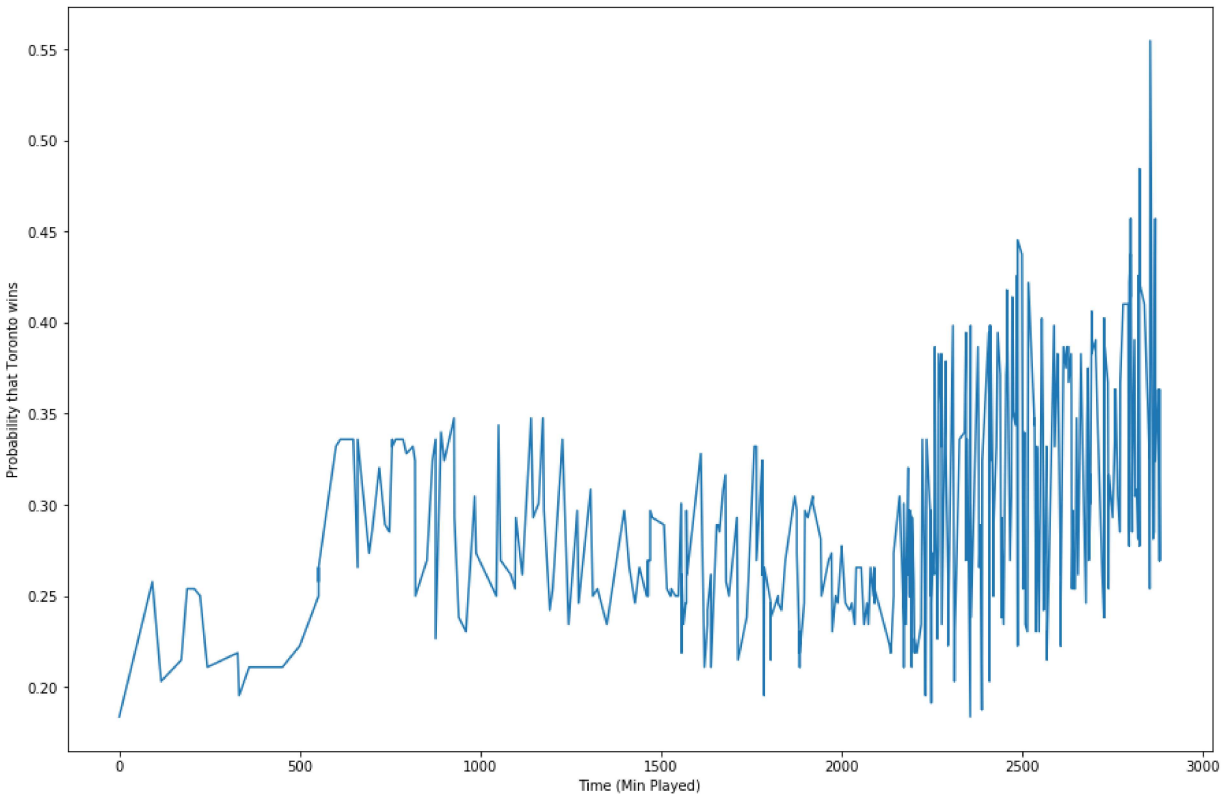
In [126]:
```
graph_game(61, 'Cleveland', rand_forest)
```



In [127]:
```
graph_game(62, 'Toronto', rand_forest)
```



In [129]:
```
graph_game(60, 'Toronto', rand_forest)
```



In [ ]:
```
def get_record(season):
    record = dict()
    for i in season['game_id'].unique():
        rows = season.loc[season['game_id'] == i]
        row = rows.iloc[2]
        if row['awayteam'] == 'Boston':
            if row['isawayevent'] == row['win']:
                record[i] = 'away win'
            else:
                record[i] = 'away loss'
        else:
            if row['isawayevent'] != row['win']:
                record[i] = 'home win'
            else:
                record[i] = 'home loss'
    return record
```

In [ ]:
```
C_record = get_record(season_df)
```

In [ ]:
```
fig = plt.figure(figsize = (15, 10))
loss_scores = []
win_scores = []
for k, v in C_record.items():
    if v == 'away loss':
        game = season_df.loc[season_df['game_id'] == k]
        loss_scores.append(np.max(game['awayscore']))
    elif v == 'home loss':
        game = season_df.loc[season_df['game_id'] == k]
        loss_scores.append(np.max(game['homescore']))
    elif v == 'away win':
        game = season_df.loc[season_df['game_id'] == k]
        win_scores.append(np.max(game['awayscore']))
    else:
        game = season_df.loc[season_df['game_id'] == k]
        win_scores.append(np.max(game['homescore']))
plt.subplot(2,1,1)
plt.hist(loss_scores, bins = 6)
plt.ylabel("Frequency")
plt.xlabel("Final Score (Loss)")
plt.subplot(2,1,2)
plt.hist(win_scores, bins = 18)
plt.ylabel("Frequency")
plt.xlabel("Final Score (Win)")
plt.show()
```

In [ ]:
```
fig = plt.figure(figsize = (15, 10))
poss_loss = []
poss_win = []
for k, v in C_record.items():
    if v == 'away loss':
        game = season_df.loc[season_df['game_id'] == k]
        x = game['isawayevent'].value_counts()
        poss_loss.append((x.values[0]/np.sum(x.values)))
    elif v == 'home loss':
        game = season_df.loc[season_df['game_id'] == k]
        x = game['isawayevent'].value_counts()
        poss_loss.append((x.values[1]/np.sum(x.values)))
    if v == 'away win':
        game = season_df.loc[season_df['game_id'] == k]
        x = game['isawayevent'].value_counts()
        poss_win.append((x.values[0]/np.sum(x.values)))
    elif v == 'home win':
        game = season_df.loc[season_df['game_id'] == k]
        x = game['isawayevent'].value_counts()
        poss_win.append((x.values[1]/np.sum(x.values)))

plt.subplot(2,1,1)
plt.hist(poss_loss, bins = 10)
plt.ylabel("Frequency")
plt.xlabel("Possesion (Loss)")
plt.subplot(2,1,2)
plt.hist(poss_win, bins = 18)
plt.ylabel("Frequency")
plt.xlabel("Possesion (Win)")
plt.show()
```

In [ ]:
```
poss = []
for k, v in C_record.items():
    if v == 'away loss':
        game = season_df.loc[season_df['game_id'] == k]
        x = game['isawayevent'].value_counts()
        poss.append((x.values[0]/np.sum(x.values)))
    elif v == 'home loss':
        game = season_df.loc[season_df['game_id'] == k]
        x = game['isawayevent'].value_counts()
        poss.append((x.values[1]/np.sum(x.values)))
    if v == 'away win':
        game = season_df.loc[season_df['game_id'] == k]
        x = game['isawayevent'].value_counts()
        poss.append((x.values[0]/np.sum(x.values)))
    elif v == 'home win':
        game = season_df.loc[season_df['game_id'] == k]
        x = game['isawayevent'].value_counts()
        poss.append((x.values[1]/np.sum(x.values)))

plt.plot(range(22), poss)
plt.xlabel("Game")
plt.ylabel("Possesion")
plt.show()
```

In [ ]:
```
# setting 1 if rebound included else 0
reb = []
for i in pbp4['event']:
    if 'rebound' in i:
        reb.append(1)
    else:
        reb.append(0)
pbp4['is_rebound'] = reb
pbp4.head()
```

In [ ]:
```
# finding player that did the offensive rebound
vals = []
off_reb = []
for i in pbp4['event']:
    if 'Offensive rebound' in i:
        vals.append(i)
        uff = i.split('by', 1)[1]
        off_reb.append(uff)
    else:
        off_reb.append("")
pbp4['offensive_rebound'] = off_reb
pbp4.head()

# finding if away team did the offensive rebound
pbp['awayevents']
pbbp = pbp[pd.notnull(pbp['awayevents'])]
away_vals = pbbp['awayevents'].values

away_match = []
for i in pbp4['event']:
    if 'Offensive rebound' in i:
        if i in away_vals:
            away_match.append(1)
        else:
            away_match.append(0)
    else:
        away_match.append(0)

pbp4['away_offensive_rebound'] = away_match

# finding if home team did the offensive rebound
pbp['homeevents']
pbbbp = pbp[pd.notnull(pbp['homeevents'])]
home_vals = pbbbp['homeevents'].values

home_match = []
for i in pbp4['event']:
    if 'Offensive rebound' in i:
        if i in home_vals:
            home_match.append(1)
        else:
            home_match.append(0)
    else:
        home_match.append(0)

pbp4['home_offensive_rebound'] = home_match
pbp4.head()
```

In [ ]:
```
# finding player that did the defensive rebound
def_reb = []
for i in pbp4['event']:
    if 'Defensive rebound' in i:
        uf = i.split('by', 1)[1]
        def_reb.append(uf)
    else:
        def_reb.append("")
pbp4['defensive_rebound'] = def_reb

# finding if away team did the defensive rebound
away_match1 = []
for i in pbp4['event']:
    if 'Defensive rebound' in i:
        if i in away_vals:
            away_match1.append(1)
        else:
            away_match1.append(0)
    else:
        away_match1.append(0)

pbp4['away_defensive_rebound'] = away_match1

# finding if home team did the defensive rebound
home_match1 = []
for i in pbp4['event']:
    if 'Defensive rebound' in i:
        if i in home_vals:
            home_match1.append(1)
        else:
            home_match1.append(0)
    else:
        home_match1.append(0)

pbp4['home_defensive_rebound'] = home_match1
pbp4
```

In [ ]:
```
st = "J. Redick misses 3-pt shot from 25 ft"
ai = st.split('-pt', 1)[0]
points = ai.split('misses ',1)[1]
ai1 = st.split('from ', 1)[1]
feet = ai1.split(' ft', 1)[0]
player = st.split(' misses', 1)[0]
```

In [ ]:
```
# miss shot events
miss_points = []
miss_feet = []
miss_player = []
for i in pbp4['event']:
    if 'misses' in i:
        st = "J. Redick misses 3-pt shot from 25 ft"
        ai = st.split('-pt', 1)[0]
        points = ai.split('misses ',1)[1]
        ai1 = st.split('from ', 1)[1]
        feet = ai1.split(' ft', 1)[0]
        player = st.split(' misses', 1)[0]
        miss_points.append(points)
        miss_feet.append(feet)
        miss_player.append(player)
    else:
        miss_points.append(0)
        miss_feet.append(0)
        miss_player.append(0)

pbp4['miss_points'] = miss_points
pbp4['miss_feet'] = miss_feet
pbp4['miss_player'] = miss_player
pbp4.head()
```

In [ ]:
```
# finding players that missed 3-pt shot from 25 ft
p325 = []
for i in pbp4['event']:
    if 'misses 3-pt shot from 25 ft' in i:
        pp = i.split(' misses', 1)[0]
        p325.append(pp)
    else:
        p325.append("")
pbp4['miss_3_25'] = p325

# finding if away players missed 3-pt shot from 25 ft
away_match2 = []
for i in pbp4['event']:
    if 'misses 3-pt shot from 25 ft' in i:
        if i in away_vals:
            away_match2.append(1)
        else:
            away_match2.append(0)
    else:
        away_match2.append(0)
pbp4['away_miss_3_25'] = away_match2

# finding if home players missed 3-pt shot from 25 ft
home_match2 = []
for i in pbp4['event']:
    if 'misses 3-pt shot from 25 ft' in i:
        if i in home_vals:
            home_match2.append(1)
        else:
            home_match2.append(0)
    else:
        home
        home_match2.append(0)
pbp4['home_miss_3_25'] = home_match2
pbp4.head()
```

In [ ]:
```
# finding players that missed 2-pt shot from 14 ft
p214 = []
for i in pbp4['event']:
    if 'misses 2-pt shot from 14 ft' in i:
        ppp = i.split(' misses', 1)[0]
        p214.append(ppp)
    else:
        p214.append("")
pbp4['miss_2_14'] = p214

# finding if away players missed 2-pt shot from 14 ft
away_match3 = []
for i in pbp4['event']:
    if 'misses 2-pt shot from 14 ft' in i:
        if i in away_vals:
            away_match3.append(1)
        else:
            away_match3.append(0)
    else:
        away_match3.append(0)
pbp4['away_miss_2_14'] = away_match3

# finding if home players missed 2-pt shot from 14 ft
home_match3 = []
for i in pbp4['event']:
    if 'misses 2-pt shot from 14 ft' in i:
        if i in home_vals:
            home_match3.append(1)
        else:
            home_match3.append(0)
    else:
        home
        home_match3.append(0)
pbp4['home_miss_2_14'] = home_match3
```

In [ ]:
```
#should create lots of variables, like:
#quarter (how to treat overtime?)
#isassist
#assistplayer
#shooter
#is3ptshot...
```

In [ ]:
```
url = 'https://www.basketball-reference.com/leagues/NBA_2018_games-october.html'
response = requests.get(url)

soup = BeautifulSoup(response.text, 'lxml')
links = []
for ref in soup.find_all('a'):
    link = ref.get('href')
    if link.startswith('/boxscores/2'):
        links.append(link)
print(len(links))
links[0:2]

urls = []
for link in links:
    urls.append('https://www.basketball-reference.com' + link)
print(urls[0:2])
```

In [ ]:
```
req = requests.get("https://www.basketball-reference.com/leagues/NBA_2018_games-o
page = req.text
soup = BeautifulSoup(page, 'html.parser')
rows = soup.find_all('tr')
tables = soup.find_all('table')
months = ['october', 'november']
len(tables)
```

In [ ]:
```

```

In [ ]:
```
schedule_BOS = team_schedule('Boston Celtics', months)
```

In [ ]:
```
def to_pbp_url(game_id):
    return "https://www.basketball-reference.com/boxscores/pbp/" + game_id
```

In [ ]:
```
BOS_pbp_urls = list(map(to_pbp_url, schedule_BOS))
BOS_pbp_urls
```

In [ ]:
```

```