# playoffs

March 25, 2018

In [4]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import time
%matplotlib inline
```

In [2]:
```
from nba_py.team import TeamYearOverYearSplits, TeamList
team_list = TeamList().info().head(30)
```

In [51]:
```
rockets = TeamYearOverYearSplits(1610612745).by_year()
rockets.head(1)
```

Out[51]:

| | GROUP_SET | GROUP_VALUE | GP | W | L | W_PCT | MIN | FGM | FGA | FG_PCT | \ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | By Year | 2017-18 | 71 | 57 | 14 | 0.803 | 48.1 | 39.0 | 84.1 | 0.463 | |

| | ... | TOV_RANK | STL_RANK | BLK_RANK | BLKA_RANK | PF_RANK | PFD_RANK | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | ... | 3 | 3 | 11 | 6 | 7 | 7 | |

| | PTS_RANK | PLUS_MINUS_RANK | CFID | CFPARAMS |
|---|---|---|---|---|
| 0 | 2 | 1 | 210 | 2017-18 |

[1 rows x 56 columns]

In [3]:
```
season_team = {}
for team in team_list['TEAM_ID']:
    df = TeamYearOverYearSplits(team,season_type='Playoffs').by_year()
    for index, row in df.iterrows():
        season_data =  season_team.get(row['GROUP_VALUE'])
        if season_data:
            if team not in season_team[row['GROUP_VALUE']]:
                season_team[row['GROUP_VALUE']].append(team)
        else:
            season_team[row['GROUP_VALUE']] = [team]
    time.sleep(2)
```

In [5]:
```
def playoff_team(team_id, season):
    if team_id in season_team[season]:
```

```
            return 1
        return 0
```

```
In [24]: all_team_data = pd.DataFrame()
         for team in team_list['TEAM_ID']:
             team_data = TeamYearOverYearSplits(team,measure_type = 'Advanced').by_year()
             team_data['PLAYOFFS'] = team_data.apply(lambda row: playoff_team(team,row['GROUP_
             all_team_data = pd.concat([all_team_data,team_data])
             time.sleep(2)
```

```
In [14]: all_team_data = pd.DataFrame()
         for team in team_list['TEAM_ID']:
             team_data = TeamYearOverYearSplits(team).by_year()
             team_data['PLAYOFFS'] = team_data.apply(lambda row: playoff_team(team,row['GROUP_
             all_team_data = pd.concat([all_team_data,team_data])
             time.sleep(2)
```

```
In [6]: regular_stats = pd.read_csv('all_team_playoffs.csv')
        advs_stats = pd.read_csv('all_team_playoffs_adv.csv')
```

```
In [7]: regular_features = regular_stats[['FGM', 'FGA', 'FG_PCT', 'FG3M', 'FG3A', 'FG3_PCT', '
              'FT_PCT', 'OREB', 'DREB', 'REB', 'AST', 'TOV', 'STL', 'BLK', 'BLKA',
              'PF', 'PFD', 'PTS', 'PLUS_MINUS']]
```

```
In [8]: advs_features = advs_stats[['NET_RATING', 'AST_PCT', 'AST_TO',
              'AST_RATIO', 'OREB_PCT', 'DREB_PCT', 'REB_PCT', 'TM_TOV_PCT', 'EFG_PCT',
              'TS_PCT', 'PACE', 'PIE']]
```

```
In [9]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(regular_features, regular_stats['P
```

```
In [10]: from sklearn.linear_model import LogisticRegression
         regular_model = LogisticRegression()
         regular_model.fit(X_train, y_train)
         predictions = regular_model.predict(X_test)
         from sklearn.metrics import classification_report
         print(classification_report(y_test,predictions))
```

```
             precision    recall  f1-score   support

          0       0.86      0.90      0.88        87
          1       0.91      0.88      0.90       109

avg / total       0.89      0.89      0.89       196
```

```
In [11]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(advs_features, advs_stats['PLAYOF
```

In [12]:
```
adv_model = LogisticRegression()
adv_model.fit(X_train, y_train)
predictions = adv_model.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(y_test,predictions))
```

```
             precision    recall  f1-score   support

          0       0.87      0.95      0.91        96
          1       0.95      0.86      0.90       100

avg / total       0.91      0.90      0.90       196
```

In [13]:
```
test_team =  TeamYearOverYearSplits(1610612740).by_year()[['FGM', 'FGA', 'FG_PCT', 'FG
       'FT_PCT', 'OREB', 'DREB', 'REB', 'AST', 'TOV', 'STL', 'BLK', 'BLKA',
       'PF', 'PFD', 'PTS', 'PLUS_MINUS']]
logmodel.predict(test_team.head(1))
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-13-724b029c7aba> in <module>()
      2        'FT_PCT', 'OREB', 'DREB', 'REB', 'AST', 'TOV', 'STL', 'BLK', 'BLKA',
      3        'PF', 'PFD', 'PTS', 'PLUS_MINUS']]
----> 4 logmodel.predict(test_team.head(1))


NameError: name 'logmodel' is not defined
```

In [14]:
```
current_predictions_norm = {}
current_predictions_adv = {}
for index, row in team_list.iterrows():
    current = TeamYearOverYearSplits(row['TEAM_ID']).by_year()[['FGM', 'FGA', 'FG_PCT
       'FT_PCT', 'OREB', 'DREB', 'REB', 'AST', 'TOV', 'STL', 'BLK', 'BLKA',
       'PF', 'PFD', 'PTS', 'PLUS_MINUS']]
    current_adv = TeamYearOverYearSplits(row['TEAM_ID'],measure_type = 'Advanced').by
       'AST_RATIO', 'OREB_PCT', 'DREB_PCT', 'REB_PCT', 'TM_TOV_PCT', 'EFG_PCT',
       'TS_PCT', 'PACE', 'PIE']]
    current_predictions_norm[row['ABBREVIATION']] = regular_model.predict(current.hea
    current_predictions_adv[row['ABBREVIATION']] = adv_model.predict(current_adv.head
```

In [15]:
```
for norm, adv in zip(current_predictions_norm.keys(),current_predictions_adv.keys()):
    print("Normal Prediciton: "+norm+" "+str(current_predictions_norm[norm]))
    print("Advanced Prediciton: "+adv+" "+str(current_predictions_adv[adv]))
    print("------------------------------")
```

```
Normal Prediciton: ATL [0]
Advanced Prediciton: ATL [0]
------------------------------
Normal Prediciton: BOS [1]
Advanced Prediciton: BOS [1]
------------------------------
Normal Prediciton: CLE [0]
Advanced Prediciton: CLE [1]
------------------------------
Normal Prediciton: NOP [0]
Advanced Prediciton: NOP [1]
------------------------------
Normal Prediciton: CHI [0]
Advanced Prediciton: CHI [0]
------------------------------
Normal Prediciton: DAL [0]
Advanced Prediciton: DAL [0]
------------------------------
Normal Prediciton: DEN [0]
Advanced Prediciton: DEN [1]
------------------------------
Normal Prediciton: GSW [1]
Advanced Prediciton: GSW [1]
------------------------------
Normal Prediciton: HOU [1]
Advanced Prediciton: HOU [1]
------------------------------
Normal Prediciton: LAC [0]
Advanced Prediciton: LAC [1]
------------------------------
Normal Prediciton: LAL [0]
Advanced Prediciton: LAL [0]
------------------------------
Normal Prediciton: MIA [0]
Advanced Prediciton: MIA [1]
------------------------------
Normal Prediciton: MIL [0]
Advanced Prediciton: MIL [1]
------------------------------
Normal Prediciton: MIN [1]
Advanced Prediciton: MIN [1]
------------------------------
Normal Prediciton: BKN [0]
Advanced Prediciton: BKN [0]
------------------------------
Normal Prediciton: NYK [0]
Advanced Prediciton: NYK [0]
------------------------------
```

```
Normal Prediciton: ORL [0]
Advanced Prediciton: ORL [0]
------------------------------
Normal Prediciton: IND [0]
Advanced Prediciton: IND [1]
------------------------------
Normal Prediciton: PHI [1]
Advanced Prediciton: PHI [1]
------------------------------
Normal Prediciton: PHX [0]
Advanced Prediciton: PHX [0]
------------------------------
Normal Prediciton: POR [1]
Advanced Prediciton: POR [1]
------------------------------
Normal Prediciton: SAC [0]
Advanced Prediciton: SAC [0]
------------------------------
Normal Prediciton: SAS [1]
Advanced Prediciton: SAS [1]
------------------------------
Normal Prediciton: OKC [1]
Advanced Prediciton: OKC [1]
------------------------------
Normal Prediciton: TOR [1]
Advanced Prediciton: TOR [1]
------------------------------
Normal Prediciton: UTA [1]
Advanced Prediciton: UTA [1]
------------------------------
Normal Prediciton: MEM [0]
Advanced Prediciton: MEM [0]
------------------------------
Normal Prediciton: WAS [0]
Advanced Prediciton: WAS [1]
------------------------------
Normal Prediciton: DET [0]
Advanced Prediciton: DET [0]
------------------------------
Normal Prediciton: CHA [0]
Advanced Prediciton: CHA [0]
------------------------------
```

In [ ]: