```python
#importing modules
import pandas as pd
import numpy as np
import math
import datacompy
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
## Loading dataframs to analyze and cleaning datasets
sal = pd.read_csv("Salary.csv")
stats = pd.read_csv('TeamWins.csv')
stats.drop(stats.iloc[:, 4:], inplace = True, axis = 1)
stats.dropna(inplace = True)


print(sal)
print(stats)
```

```
        Year     Team  Payroll
0    2016-17  Celtics     93.5
1    2015-16  Celtics     77.1
2    2014-15  Celtics     62.2
3    2013-14  Celtics     70.0
4    2012-13  Celtics     62.6
..       ...      ...      ...
869      NaN      NaN      NaN
870      NaN      NaN      NaN
871      NaN      NaN      NaN
872      NaN      NaN      NaN
873      NaN      NaN      NaN

[874 rows x 3 columns]
        Year     Team Record  Winning Percentage
0    2016-17  Celtics  25-15               0.625
1    2015-16  Celtics  48-34               0.585
2    2014-15  Celtics  40-42               0.488
3    2013-14  Celtics  25-57               0.305
4    2012-13  Celtics  41-40               0.506
..       ...      ...    ...                 ...
503  2004-05  Wizards  45-37               0.549
504  2003-04  Wizards  25-57               0.305
505  2002-03  Wizards  37-45               0.451
506  2001-02  Wizards  37-45               0.451
507  2000-01  Wizards  19-63               0.232

[508 rows x 4 columns]
```

In the previous cell, we have just gotten our data loaded into a dataframe using pandas. From here, we can go on to interpretate the date to see if the winning percentage has a correlation with the millions in pay roll a team is spending. In this cell below, we are comparing the data based on a shared index to see if we are missing any values.

```python
compare = datacompy.Compare(sal, stats, on_index = True)
print(compare.report())
output = pd.concat([sal, stats], axis=1)
print(output)
```

```
DataComPy Comparison
--------------------

DataFrame Summary
-----------------

  DataFrame  Columns  Rows
0       df1        3   874
1       df2        4   508

Column Summary
--------------

Number of columns in common: 2
Number of columns in df1 but not in df2: 1
Number of columns in df2 but not in df1: 2

Row Summary
-----------

Matched on: index
Any duplicates on match values: No
Absolute Tolerance: 0
Relative Tolerance: 0
Number of rows in common: 508
Number of rows in df1 but not in df2: 366
Number of rows in df2 but not in df1: 0

Number of rows with some compared columns unequal: 0
Number of rows with all compared columns equal: 508

Column Comparison
-----------------

Number of columns compared with some values unequal: 0
Number of columns compared with all values equal: 2
Total number of values which compare unequal: 0

Sample Rows Only in df1 (First 10 Columns)
------------------------------------------

    year team  payroll
681  NaN  NaN      NaN
873  NaN  NaN      NaN
718  NaN  NaN      NaN
562  NaN  NaN      NaN
793  NaN  NaN      NaN
677  NaN  NaN      NaN
570  NaN  NaN      NaN
752  NaN  NaN      NaN
564  NaN  NaN      NaN
711  NaN  NaN      NaN


        year     team  payroll     year     team record  winning percentage
0    2016-17  Celtics     93.5  2016-17  Celtics  25-15               0.625
1    2015-16  Celtics     77.1  2015-16  Celtics  48-34               0.585
2    2014-15  Celtics     62.2  2014-15  Celtics  40-42               0.488
3    2013-14  Celtics     70.0  2013-14  Celtics  25-57               0.305
4    2012-13  Celtics     62.6  2012-13  Celtics  41-40               0.506
..       ...      ...      ...      ...      ...    ...                 ...
869      NaN      NaN      NaN      NaN      NaN    NaN                 NaN
870      NaN      NaN      NaN      NaN      NaN    NaN                 NaN
871      NaN      NaN      NaN      NaN      NaN    NaN                 NaN
872      NaN      NaN      NaN      NaN      NaN    NaN                 NaN
873      NaN      NaN      NaN      NaN      NaN    NaN                 NaN

[874 rows x 7 columns]
```

```python
print(output.keys())
output = output.loc[:,~output.columns.duplicated()]
output = output.iloc[0:508]
```

```
Index(['year', 'team', 'payroll', 'year', 'team', 'record',
       'winning percentage'],
      dtype='object')
```

```python
print(output.keys())
print(output)
```

```
Index(['year', 'team', 'payroll', 'record', 'winning percentage'], dtype='object')
        year     team  payroll record  winning percentage
0    2016-17  Celtics     93.5  25-15               0.625
1    2015-16  Celtics     77.1  48-34               0.585
2    2014-15  Celtics     62.2  40-42               0.488
3    2013-14  Celtics     70.0  25-57               0.305
4    2012-13  Celtics     62.6  41-40               0.506
..       ...      ...      ...    ...                 ...
503  2004-05  Wizards     60.5  45-37               0.549
504  2003-04  Wizards     52.3  25-57               0.305
505  2002-03  Wizards     53.4  37-45               0.451
506  2001-02  Wizards     51.4  37-45               0.451
507  2000-01  Wizards     50.7  19-63               0.232

[508 rows x 5 columns]
```

In the previous cell, we merged togeather the two dataframes and deleted any duplicate entries for the column headers. In the next cell, I will compute some statistcs based off of winning percentage and payroll for each team and the overall league.

```python
teams = output['team'].unique()
print(teams)
```

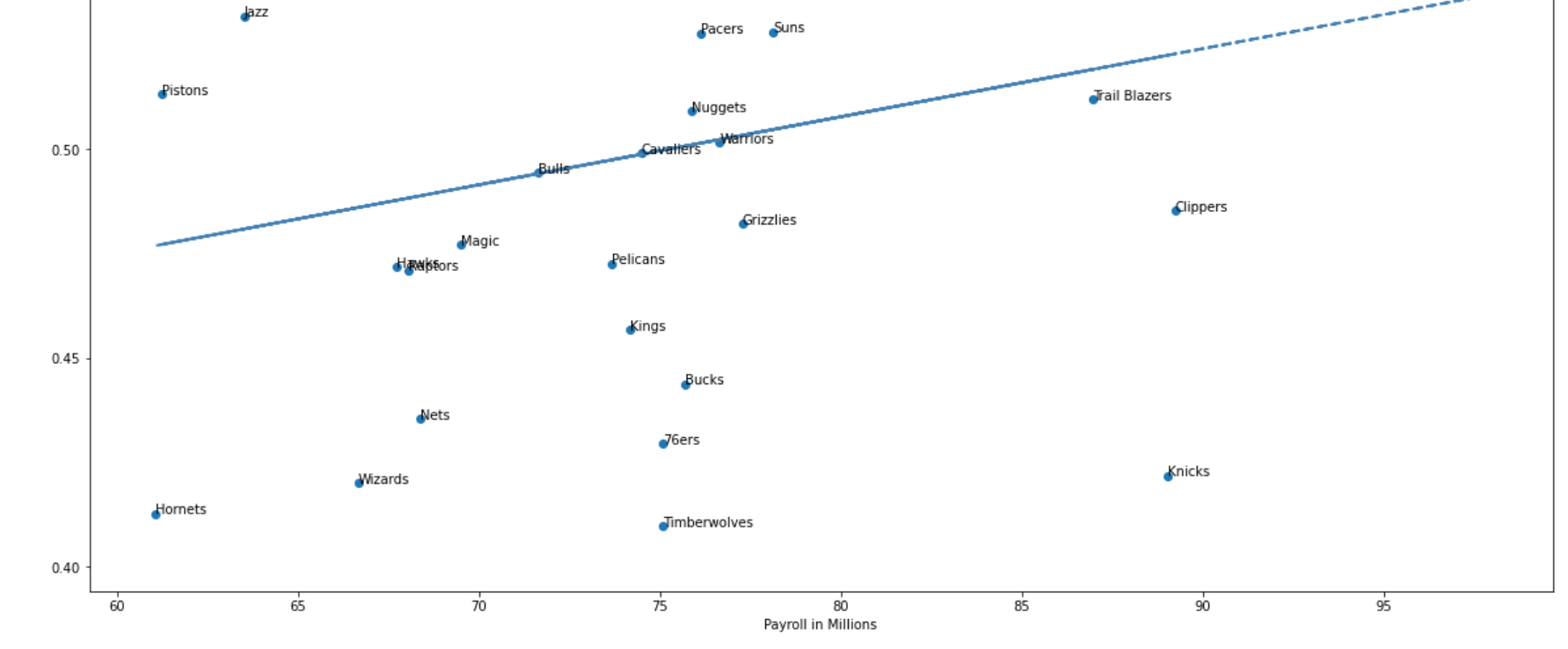
```
['Celtics' 'Hawks' 'Nets' 'Hornets' 'Bulls' 'Cavaliers' 'Mavericks'
 'Nuggets' 'Pistons' 'Warriors' 'Rockets' 'Pacers' 'Clippers' 'Lakers'
 'Grizzlies' 'Heat' 'Bucks' 'Timberwolves' 'Pelicans' 'Knicks' 'Thunder'
 'Magic' '76ers' 'Suns' 'Trail Blazers' 'Kings' 'Spurs' 'Raptors' 'Jazz'
 'Wizards']
```

```python
teams_map = {}
for i in teams:
    teams_map[i] = output.loc[output['team'] == i]
team_stat = {}

for key, value in teams_map.items():
    avgpr = value['payroll'].mean()
    avgwp = value['winning percentage'].mean()
    team_stat[key] = [avgpr, avgwp]
print("Team, Average Payroll, Winning Percentage")
totalPay = []
totalWin = []
teamList = []
for key, value in team_stat.items():
    teamList.append(key)
    totalPay.append(value[0])
    totalWin.append(value[1])
    print(key, value)
plt.figure(figsize=(20,20))
plt.scatter(totalPay, totalWin)
for i in range(len(totalPay)):
    plt.annotate(teamList[i], (totalPay[i], totalWin[i]))

plt.xlabel("Payroll in Millions")
plt.ylabel("Win Percentage")
plt.title("Payroll in Millions vs Win Percentage",fontsize=15)
AvgPR = sum(totalPay) / len(totalPay)
AvgWin =  sum(totalWin) / len(totalWin)
print(AvgPR, AvgWin)
a, b = np.polyfit(totalPay, totalWin, 1)
newTP = []
for i in range(len(totalPay)):
    newTP.append(totalPay[i] * a)
print("Coeffecient: " + str(a) , " Intercept = " + str(b))
plt.plot(totalPay, newTP+b, color='steelblue', linestyle='--', linewidth=2)
```

```
Team, Average Payroll, Winning Percentage
Celtics [65.1058823529412, 0.5418235294117647]
Hawks [67.72352941176469, 0.4718823529411765]
Nets [68.36470588235294, 0.4354117647058824]
Hornets [61.06666666666667, 0.41273333333333334]
Bulls [71.6470588235294, 0.49441176470588244]
Cavaliers [74.5, 0.49905882352941167]
Mavericks [69.8529411764706, 0.6274705882352942]
Nuggets [75.89411764705882, 0.5090588235294118]
Pistons [61.2470588235294, 0.5132941176470589]
Warriors [76.65294117647058, 0.5016470588235294]
Rockets [76.64117647058822, 0.5656470588235294]
Pacers [76.12352941176471, 0.527764705882353]
Clippers [89.24117647058823, 0.4852352941176471]
Lakers [97.83235294117647, 0.5489411764705883]
Grizzlies [77.28823529411764, 0.4822352941176471]
Heat [86.72352941176472, 0.5446470588235295]
Bucks [75.70588235294119, 0.44376470588235295]
Timberwolves [75.07647058823528, 0.40970588235294114]
Pelicans [73.66470588235293, 0.4725882352941177]
Knicks [89.0235294117647, 0.42188235294117643]
Thunder [82.57058823529412, 0.5424117647058824]
Magic [69.48235294117647, 0.4771176470588236]
76ers [75.08823529411767, 0.4294705882352941]
Suns [78.12352941176471, 0.5280588235294118]
Trail Blazers [86.98235294117646, 0.5119999999999999]
Kings [74.17058823529412, 0.4568235294117647]
Spurs [81.17647058823529, 0.719529411764706]
Raptors [68.05882352941177, 0.471]
Jazz [63.51176470588234, 0.5318823529411766]
Wizards [66.66470588235293, 0.42017647058823526]
75.17349673202612 0.4999224836601307
Coeffecient: 0.0016297329796934065  Intercept = 0.3774097568370732
```

Out[230]:

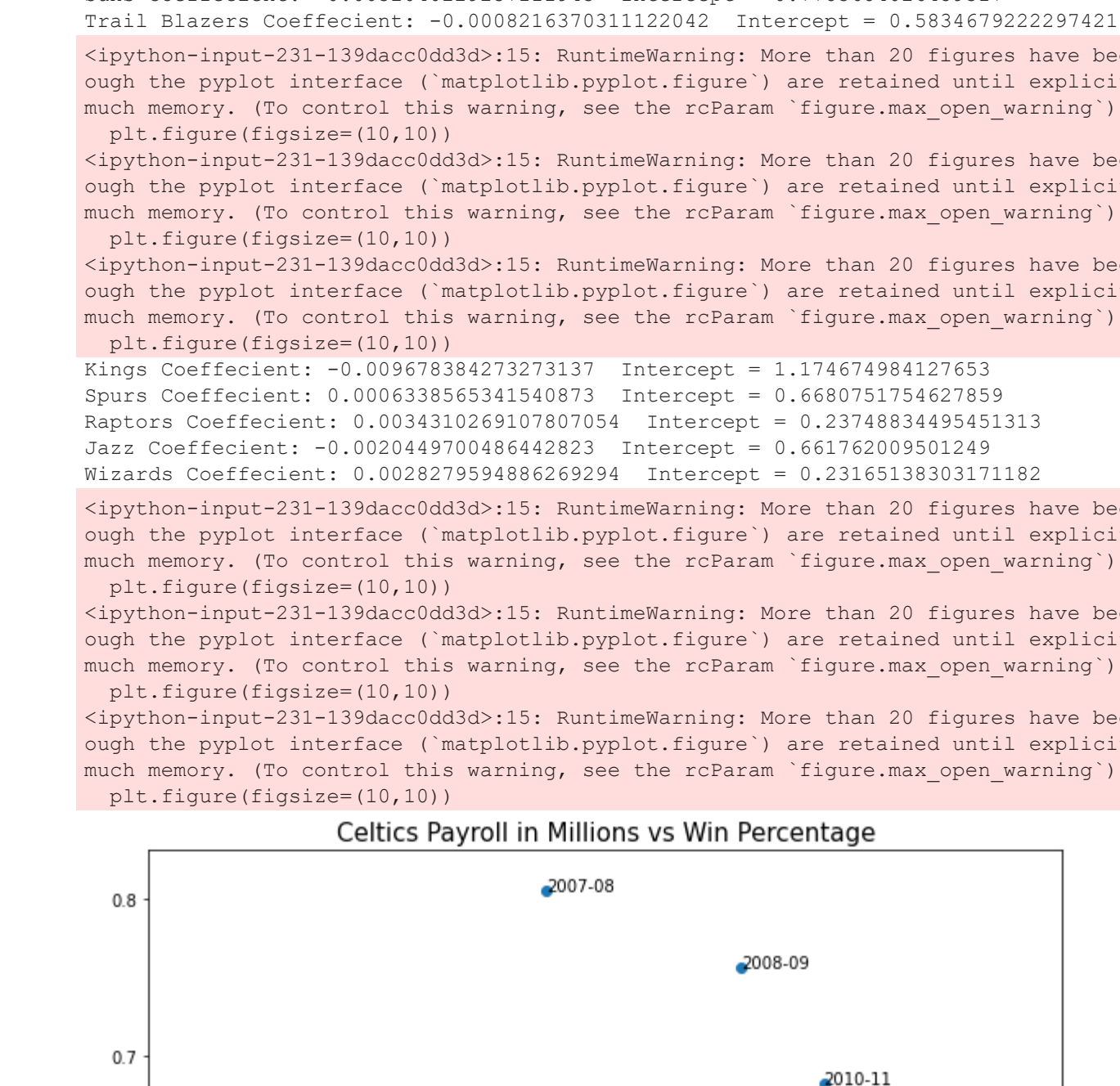
```
[<matplotlib.lines.Line2D at 0x1fa8d184ac0>]
```

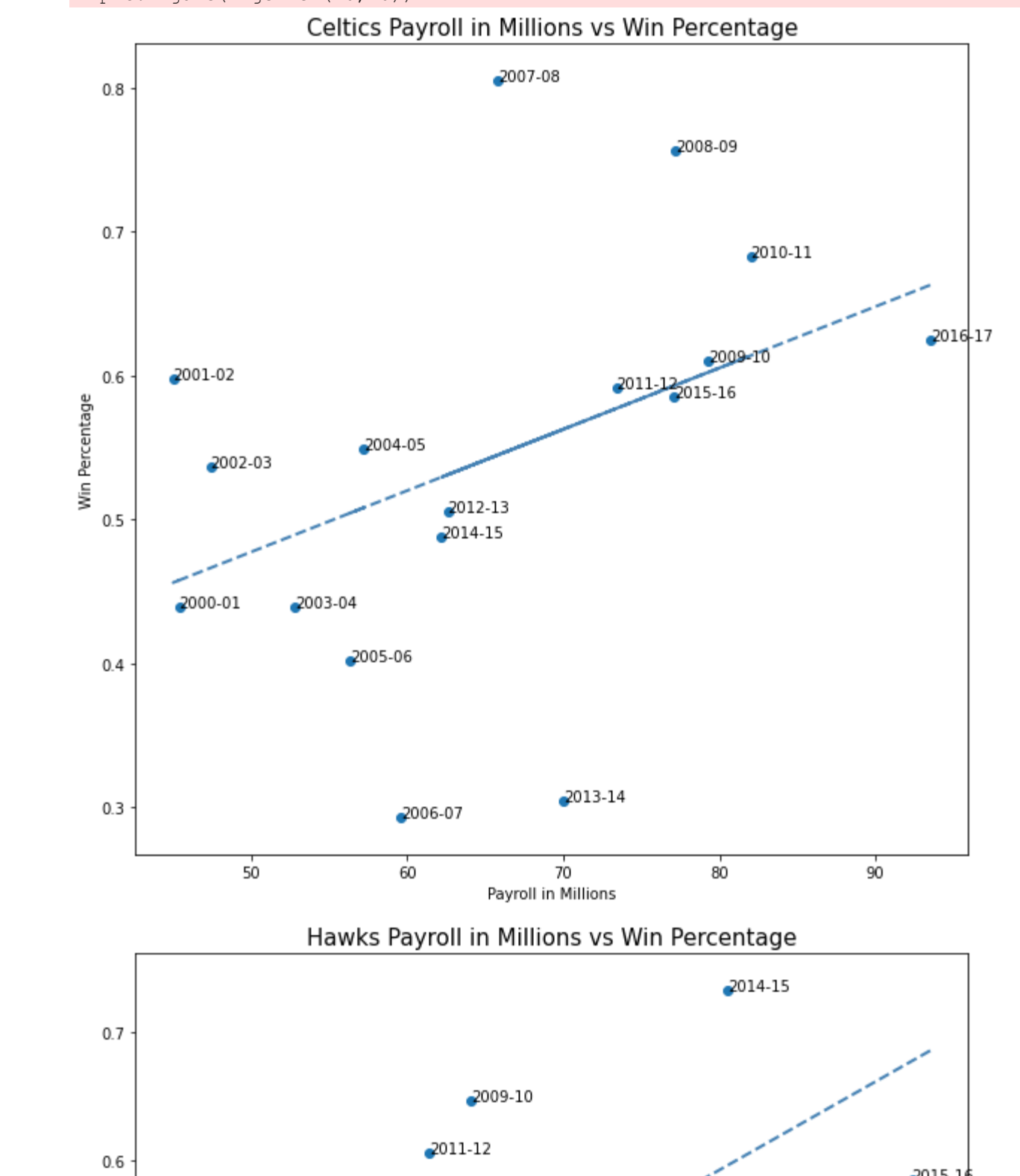


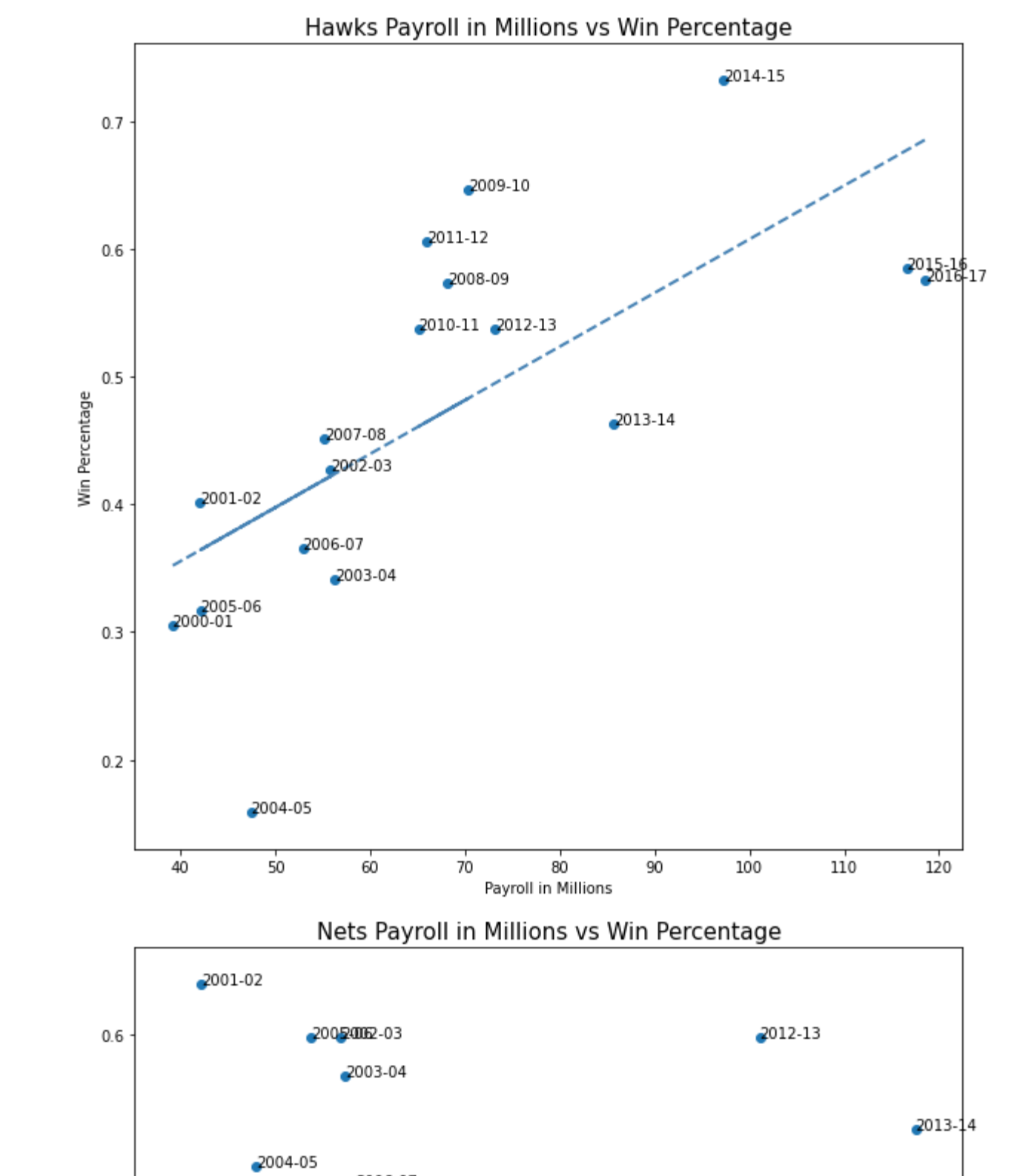
In the previous cell, we plotted the correlation of all the teams payroll in millions over the past two decades in comparision to the win percentage they are able to accomplish. Furthermore, by using simple linear regression through a * x + b, I was able to plot the line of best fit. In the following cells, I will be plotting a regression model for each teams future payroll versus wins.

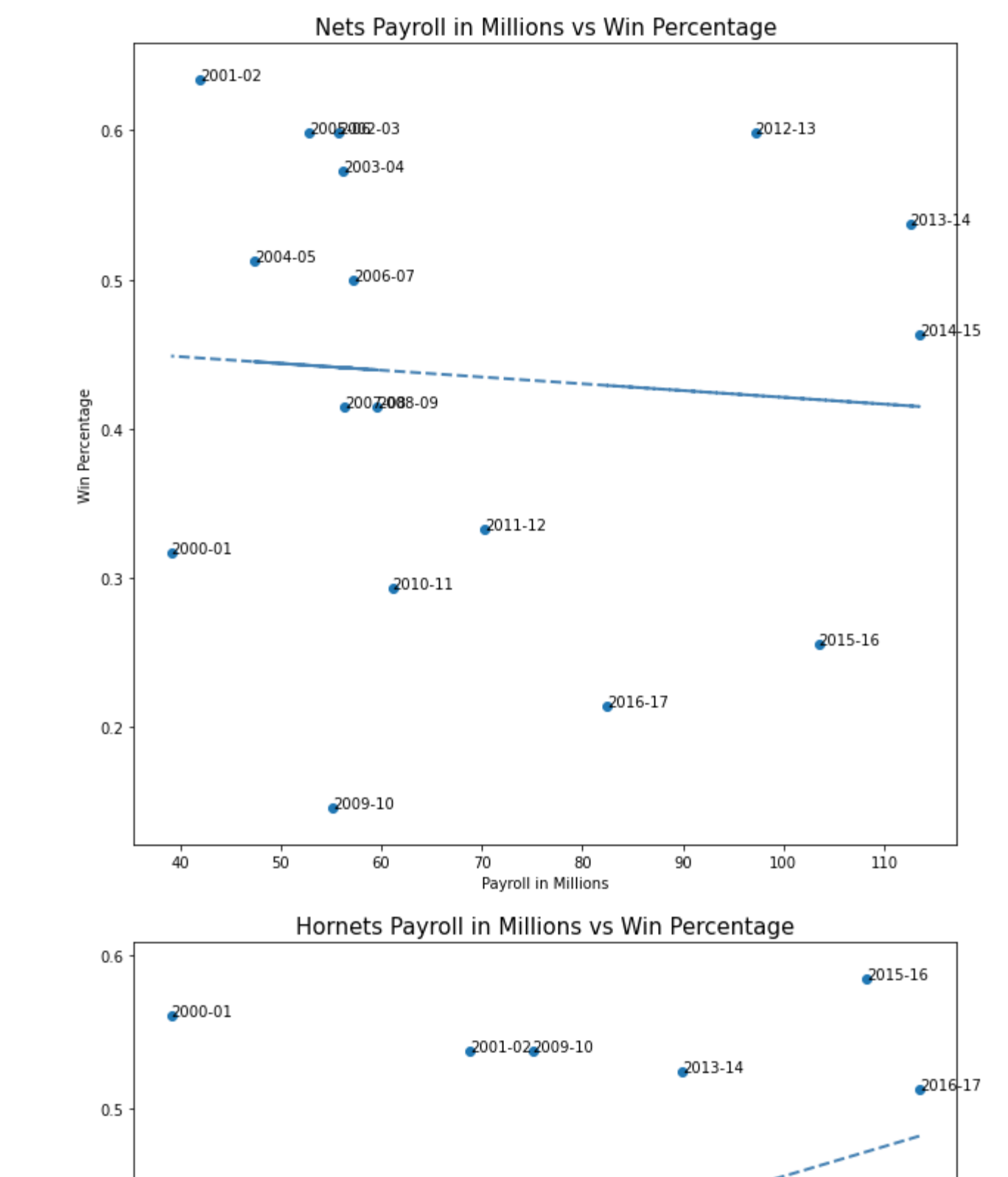
```python
from sklearn import linear_model
teamModels = {}
counter = 0
for key, value in teams_map.items():
    teamName = key
    dataFrame = value
    X = dataFrame['payroll']
    y = dataFrame['winning percentage']
    dates = dataFrame['year']
    reg = linear_model.LinearRegression()
    reg.fit(pd.DataFrame(X), y)
    co = reg.coef_[0]
    inter = reg.intercept_
    print("" + key + " Coeffecient: " + str(co) , " Intercept = " + str(inter))
    plt.figure(figsize=(10,10))
    plt.scatter(X, y)
    newTP = []
    for i in range(len(dates)):
        plt.annotate(dates[counter], (X[counter], y[counter]))
        newTP.append(co * X[counter])
        counter += 1

    plt.xlabel("Payroll in Millions")
    plt.ylabel("Win Percentage")
    plt.title( "" + teamName + " Payroll in Millions vs Win Percentage",fontsize=15)
    plt.plot(X, newTP+inter, color='steelblue', linestyle='--', linewidth=2)
```

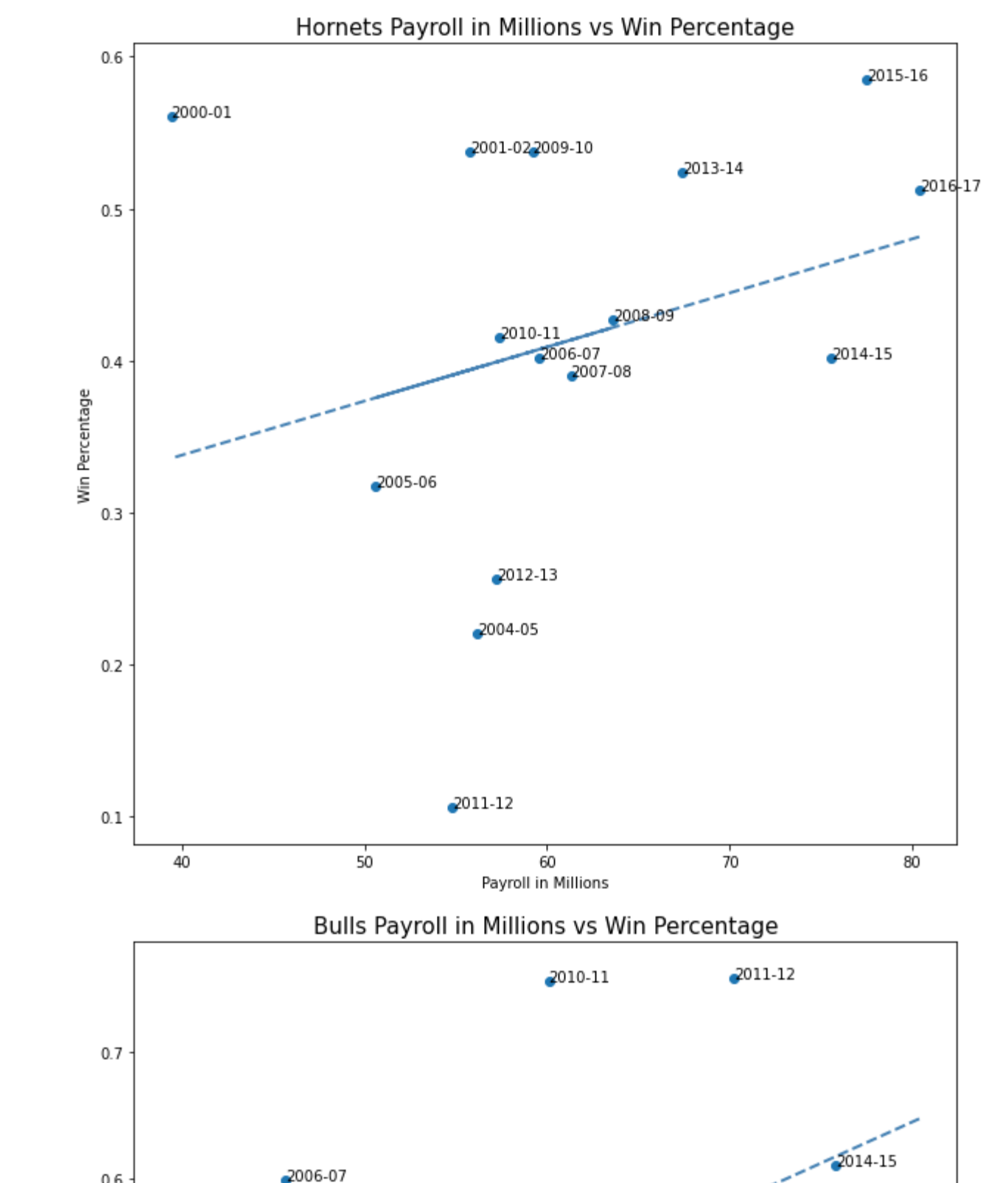
```
Celtics Coeffecient: 0.004261656643502244  Intercept = 0.26436461335127737
Hawks Coeffecient: 0.004197309475927481  Intercept = 0.18762574119792308
Nets Coeffecient: -0.00045416513482334116  Intercept = 0.4664606305700993
Hornets Coeffecient: 0.003554586113960374  Intercept = 0.19566660797415317
Bulls Coeffecient: 0.005159556919802766  Intercept = 0.12474468656942544
Cavaliers Coeffecient: -0.000502324604269234  Intercept = 0.5364820065474696
Mavericks Coeffecient: -0.004609874453578033  Intercept = 0.9494838772719951
Nuggets Coeffecient: 0.00030688676190570934  Intercept = 0.485767923517015
Pistons Coeffecient: -0.00424184267855443  Intercept = 0.7730945057006395
Warriors Coeffecient: 0.008149962903122662  Intercept = -0.12307156817994935
Rockets Coeffecient: 0.002965088590029661 7  Intercept = 0.33839918094413846
Pacers Coeffecient: -0.0004683561019293049  Intercept = 0.563417625382748
Clippers Coeffecient: 0.006789350871085185  Intercept = -0.1206543650896078
Lakers Coeffecient: -0.00722925980111219  Intercept = 1.256196672836455
Grizzlies Coeffecient: 0.007250988933409900 5  Intercept = -0.07818084468278041
Heat Coeffecient: 0.003348416248885200 5  Intercept = 0.2542605837805029
Bucks Coeffecient: -0.0018681386329276843  Intercept = 0.5851937894457606
Timberwolves Coeffecient: -0.007211471997537675  Intercept = 0.9511177476739607
Pelicans Coeffecient: -0.0032781000478479035  Intercept = 0.7140685111717606
Knicks Coeffecient: 0.0005767523671429862  Intercept = 0.3705378216215179
Thunder Coeffecient: 0.002328691868370952  Intercept = 0.3501303073157468
<ipython-input-231-139dacc0dd3d>:15: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  plt.figure(figsize=(10,10))
<ipython-input-231-139dacc0dd3d>:15: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  plt.figure(figsize=(10,10))
<ipython-input-231-139dacc0dd3d>:15: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  plt.figure(figsize=(10,10))
<ipython-input-231-139dacc0dd3d>:15: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  plt.figure(figsize=(10,10))
Magic Coeffecient: -0.002090010240282533  Intercept = 0.6223364762248076
76ers Coeffecient: -0.006987039718233969  Intercept = 0.954115070607392
Suns Coeffecient: -0.0032040229237111943  Intercept = 0.7783684026459317
Trail Blazers Coeffecient: -0.0008216370311122042  Intercept = 0.5834679222297421
<ipython-input-231-139dacc0dd3d>:15: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  plt.figure(figsize=(10,10))
<ipython-input-231-139dacc0dd3d>:15: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  plt.figure(figsize=(10,10))
<ipython-input-231-139dacc0dd3d>:15: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  plt.figure(figsize=(10,10))
Kings Coeffecient: -0.009678384273273137  Intercept = 1.174674984127653
Spurs Coeffecient: 0.0006338565341540873  Intercept = 0.6680751754627859
Raptors Coeffecient: 0.003431026910780705 4  Intercept = 0.23748834495451313
Jazz Coeffecient: -0.0020449700486442823  Intercept = 0.661762009501249
Wizards Coeffecient: 0.0028279594886269294  Intercept = 0.23165138303171182
<ipython-input-231-139dacc0dd3d>:15: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  plt.figure(figsize=(10,10))
<ipython-input-231-139dacc0dd3d>:15: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  plt.figure(figsize=(10,10))
<ipython-input-231-139dacc0dd3d>:15: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  plt.figure(figsize=(10,10))
```

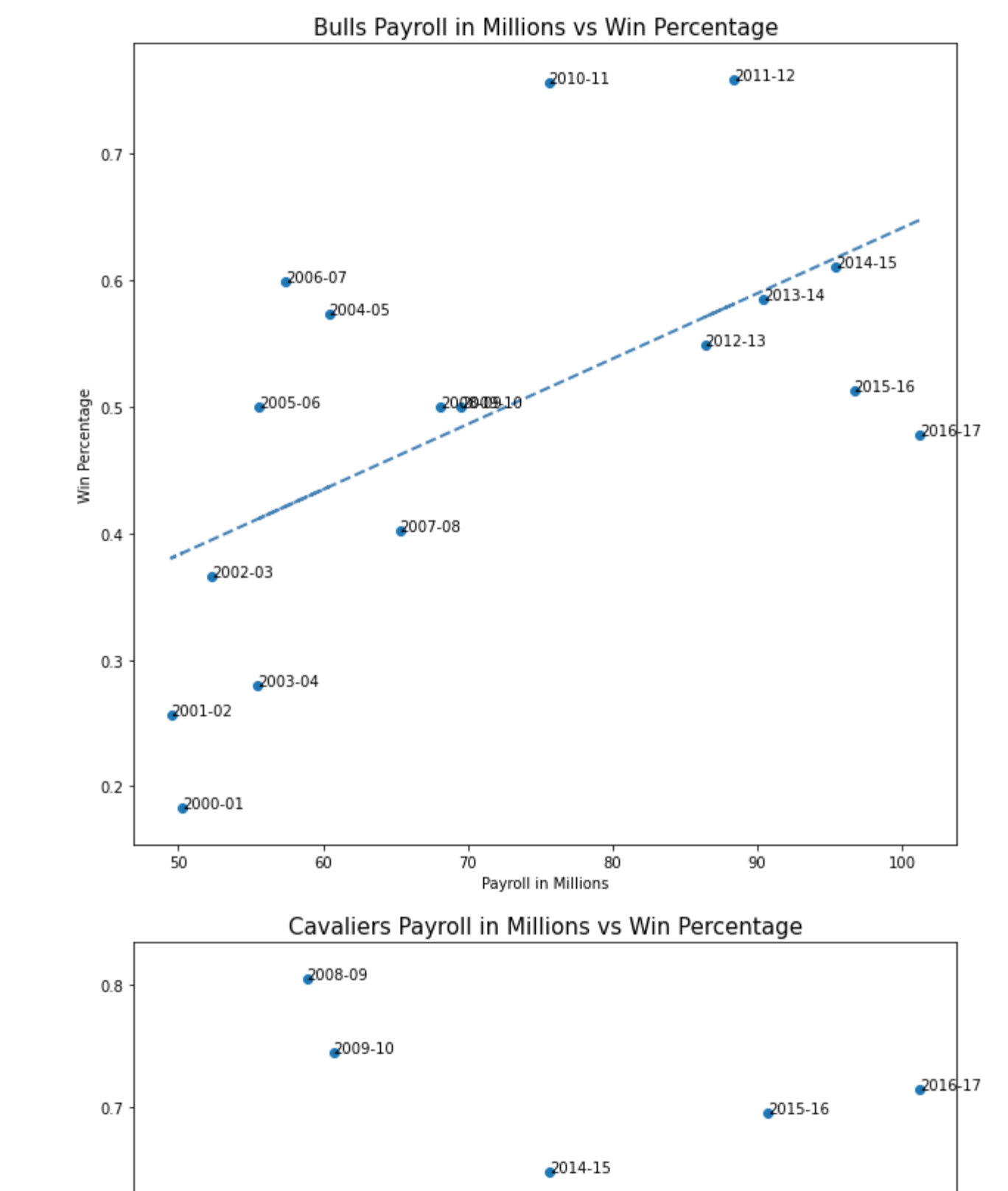


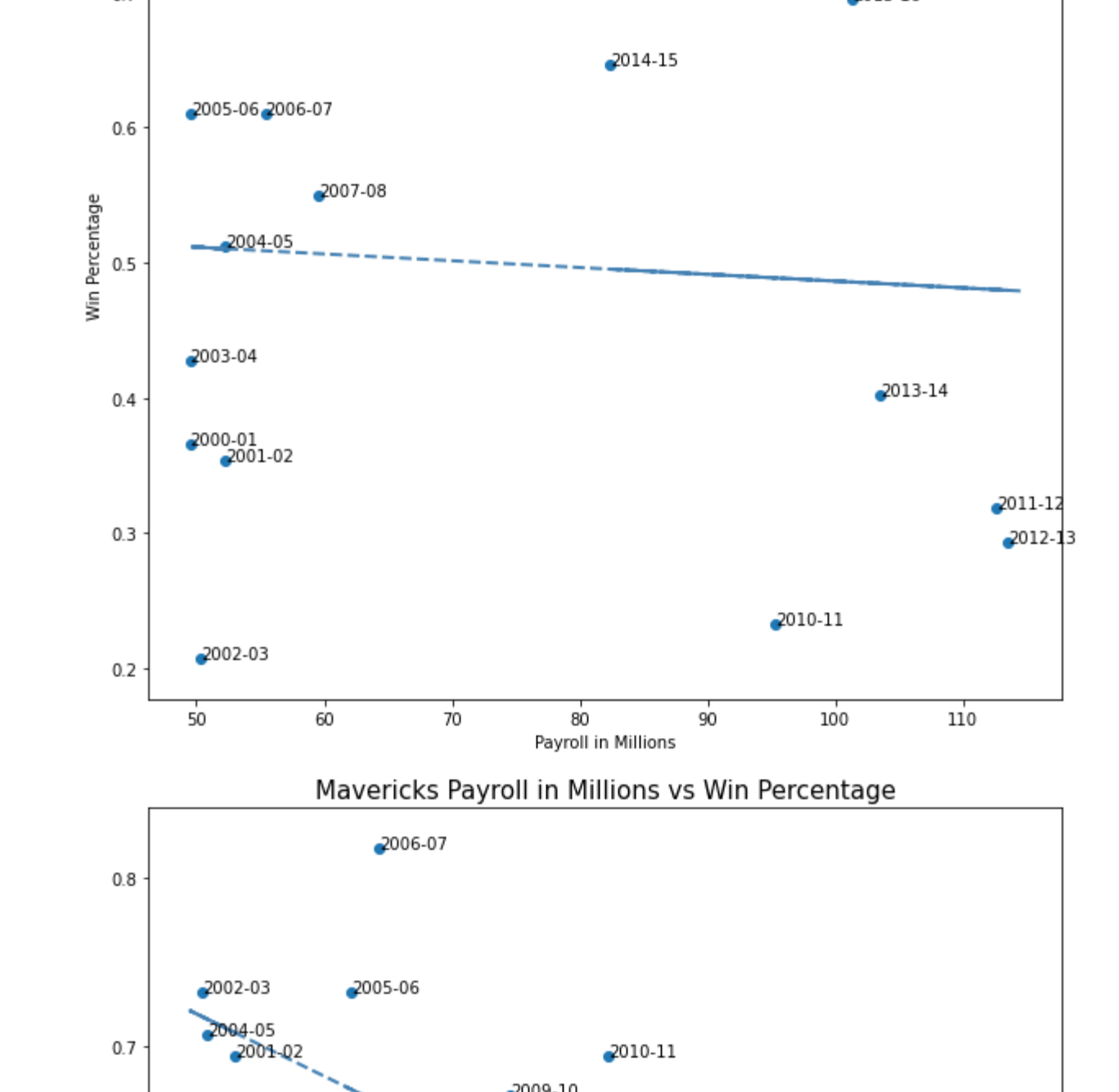


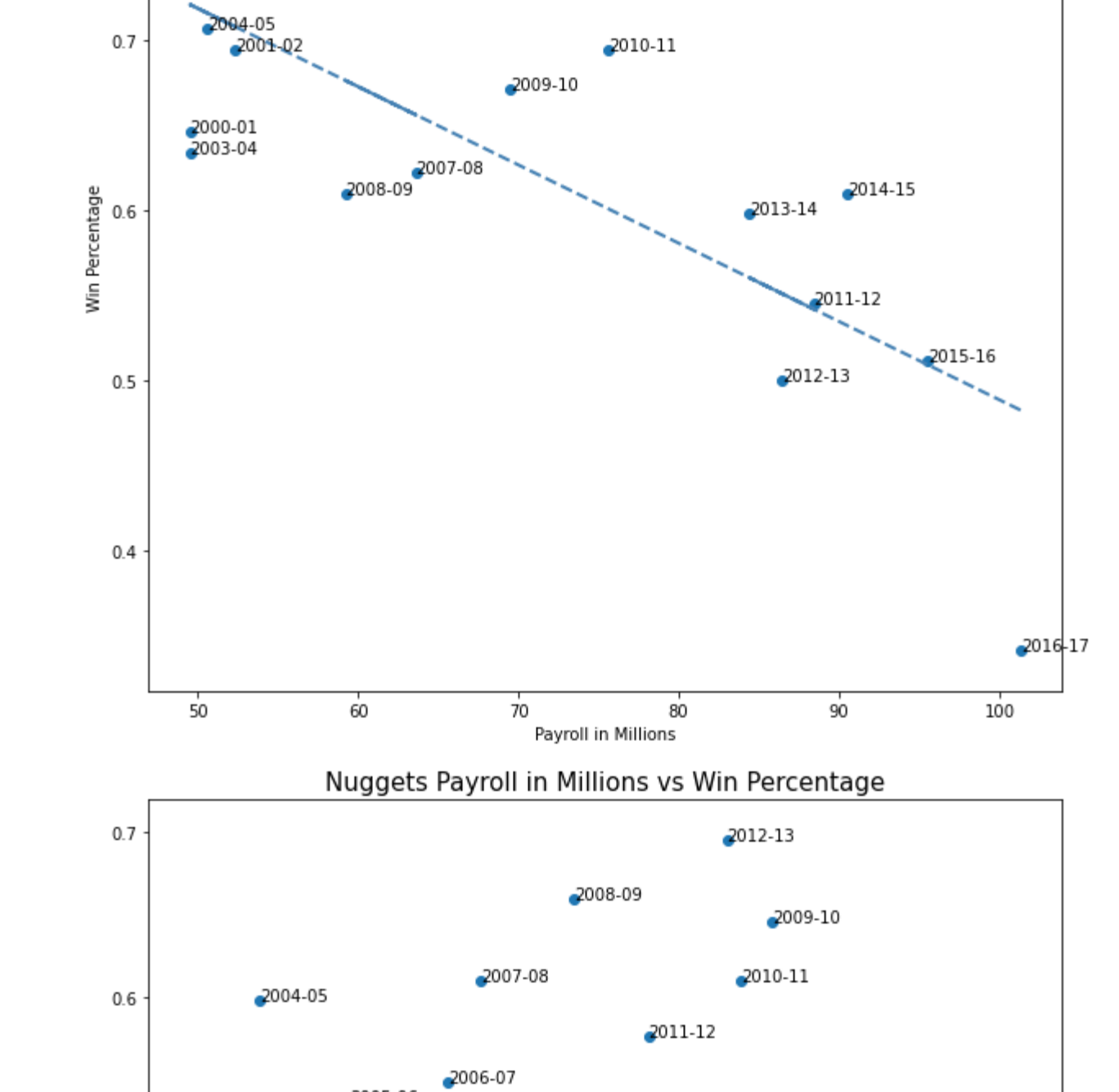


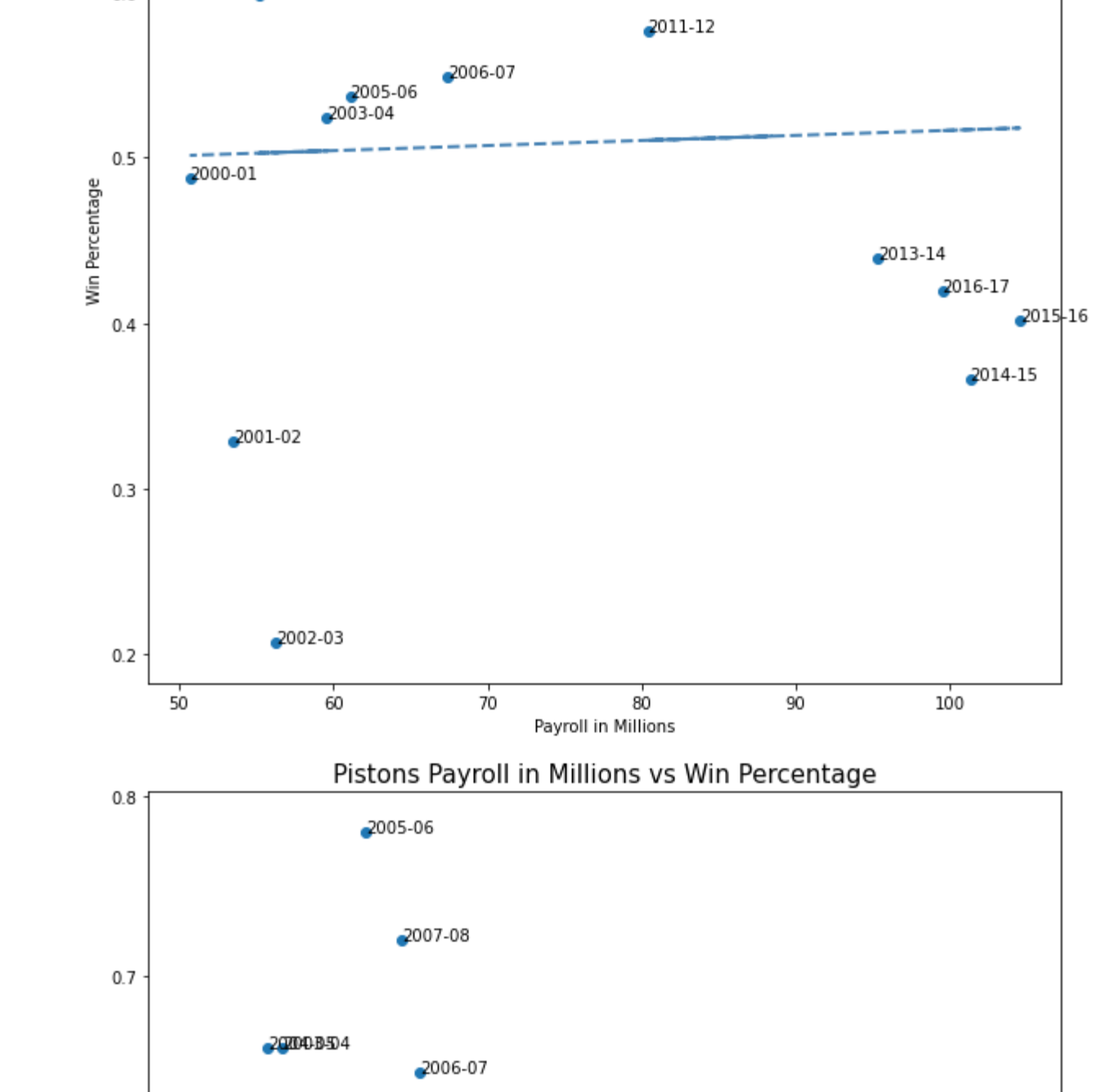


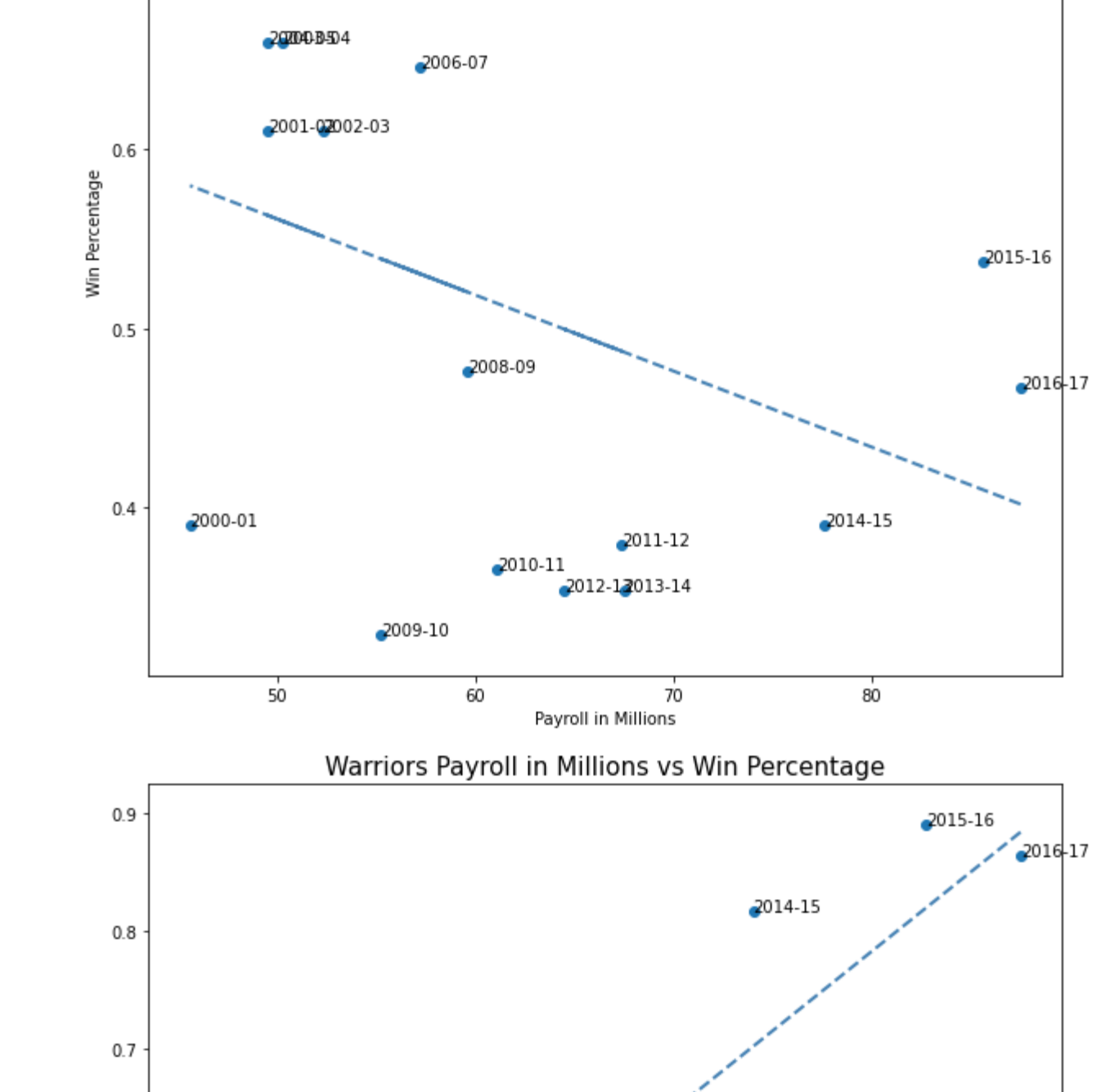


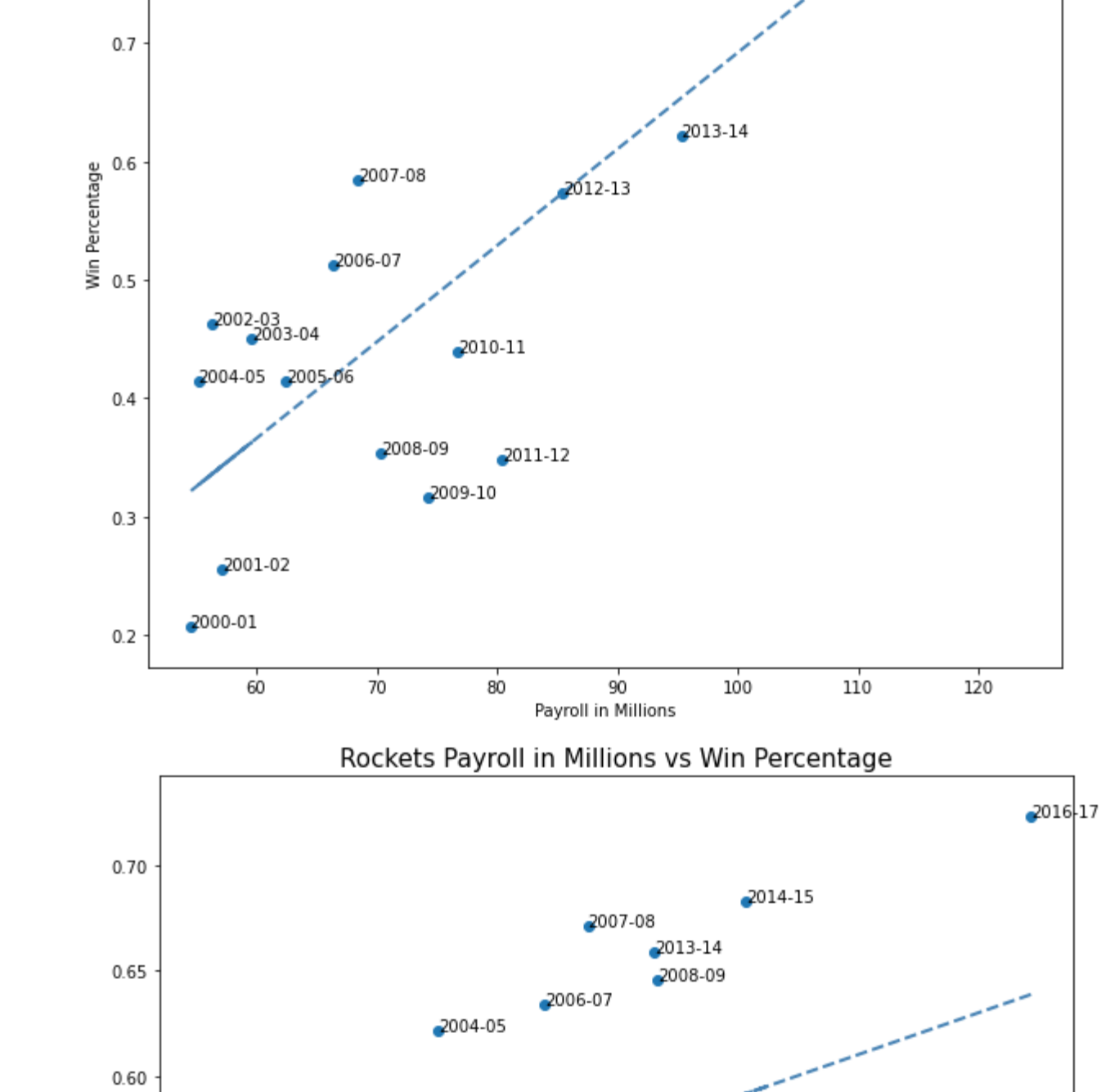


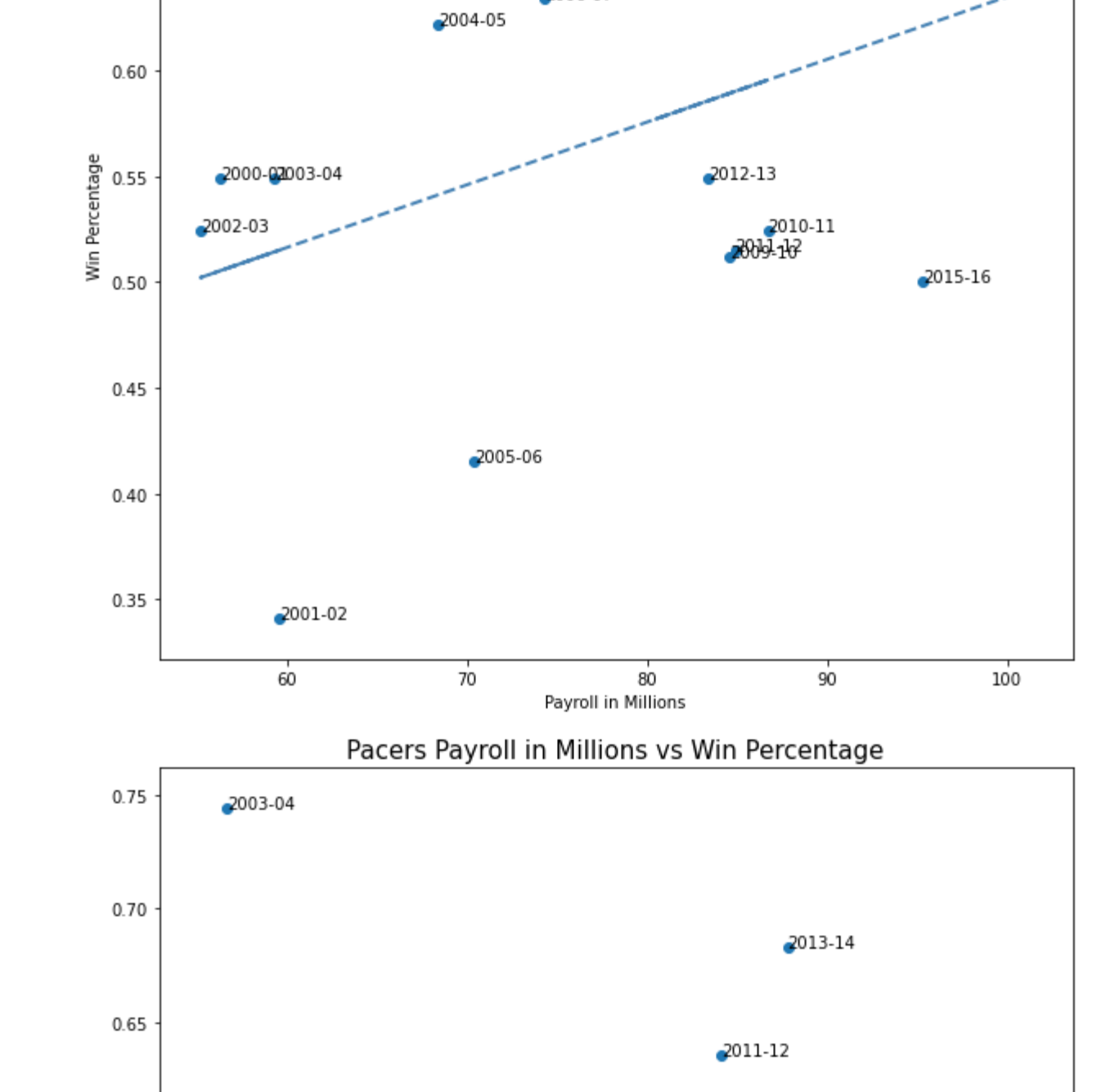


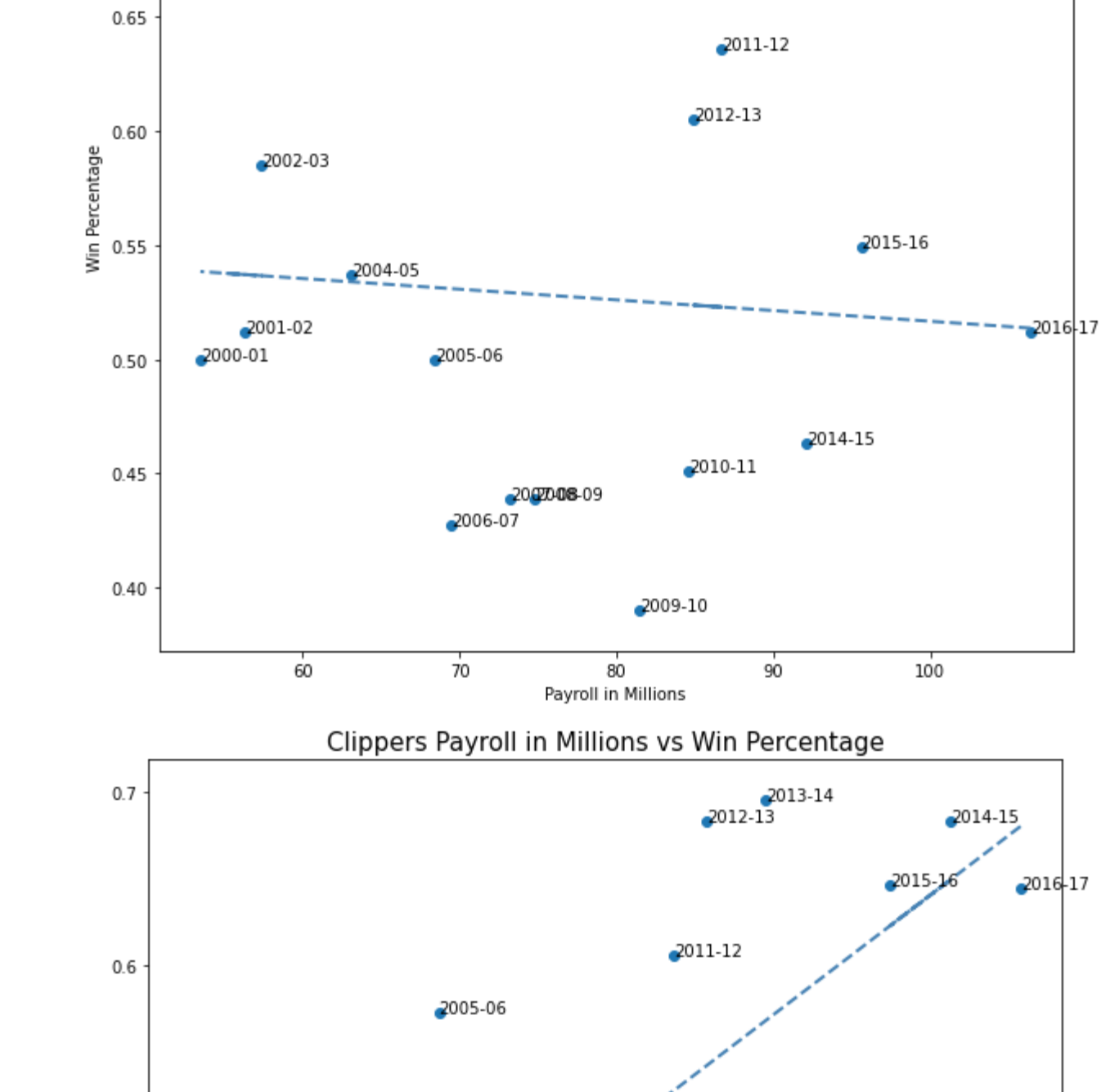


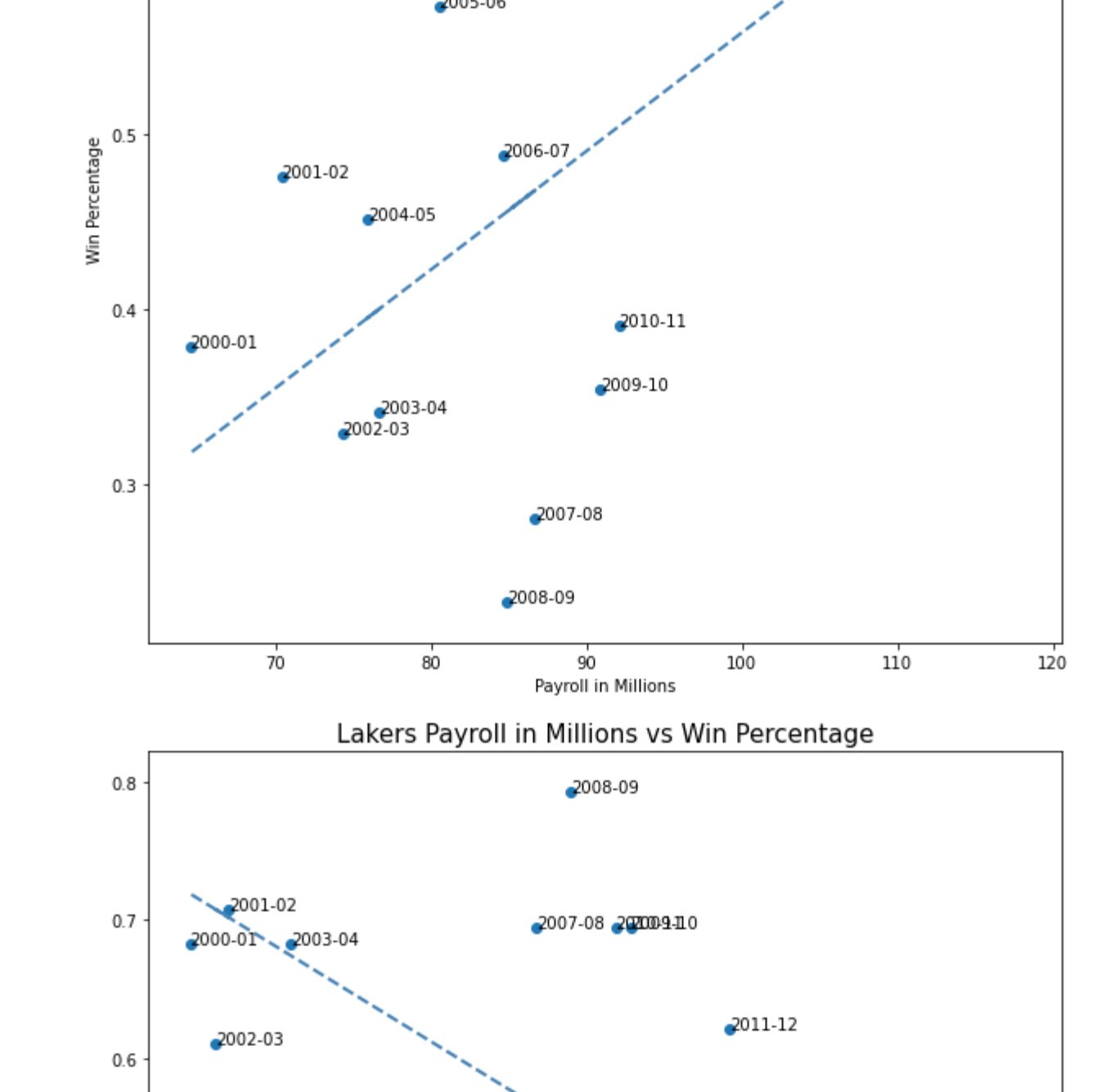


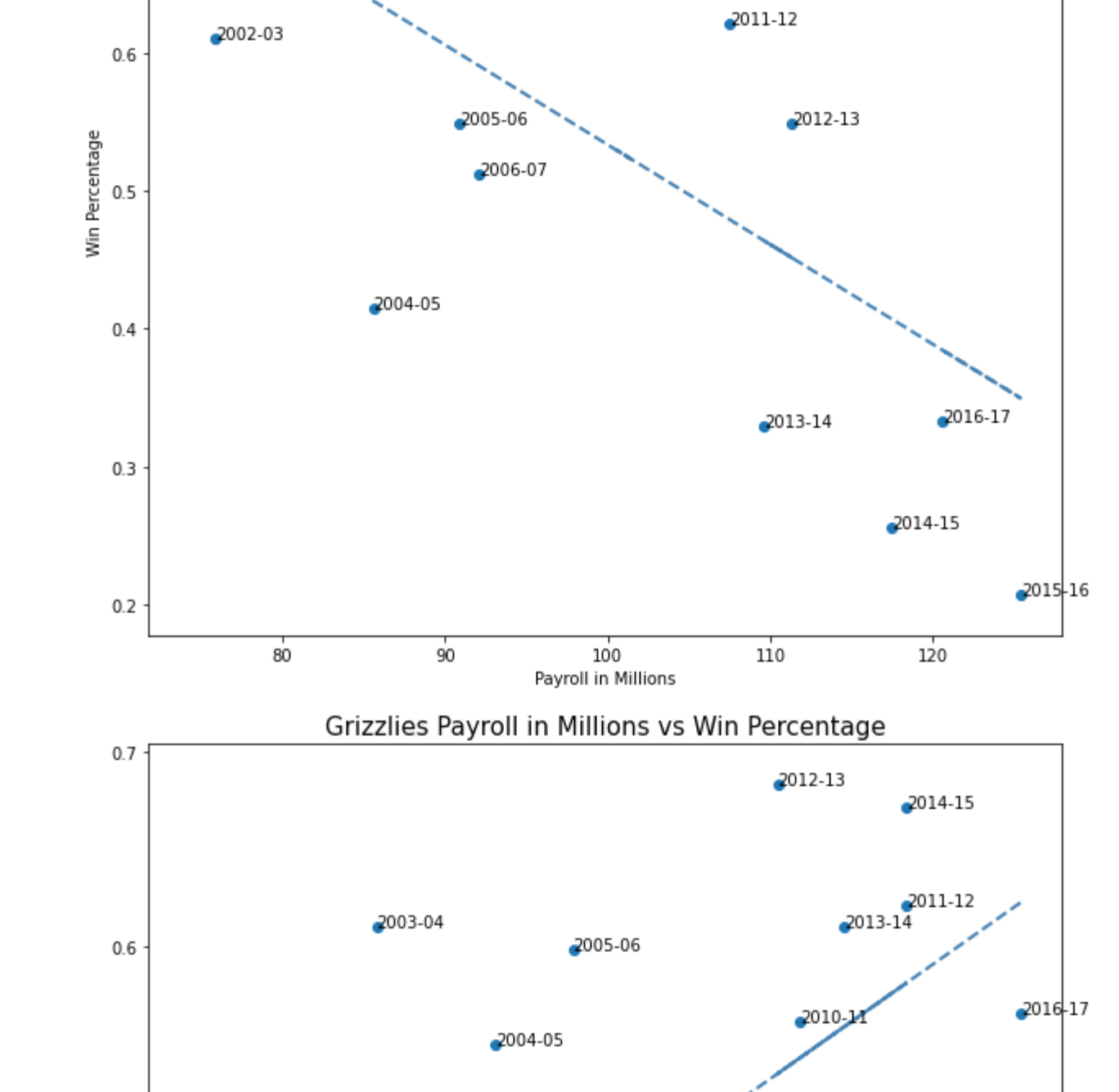


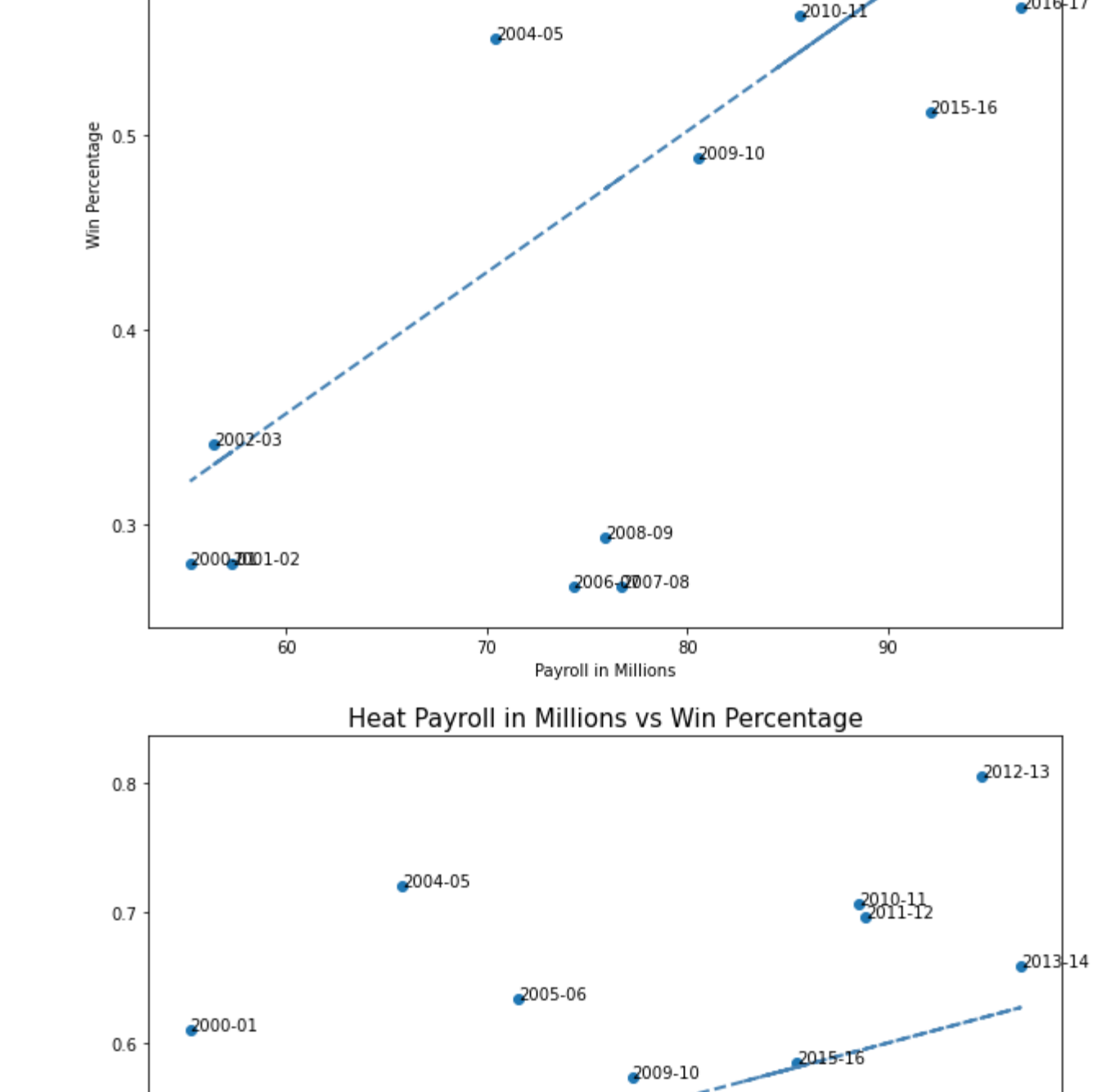


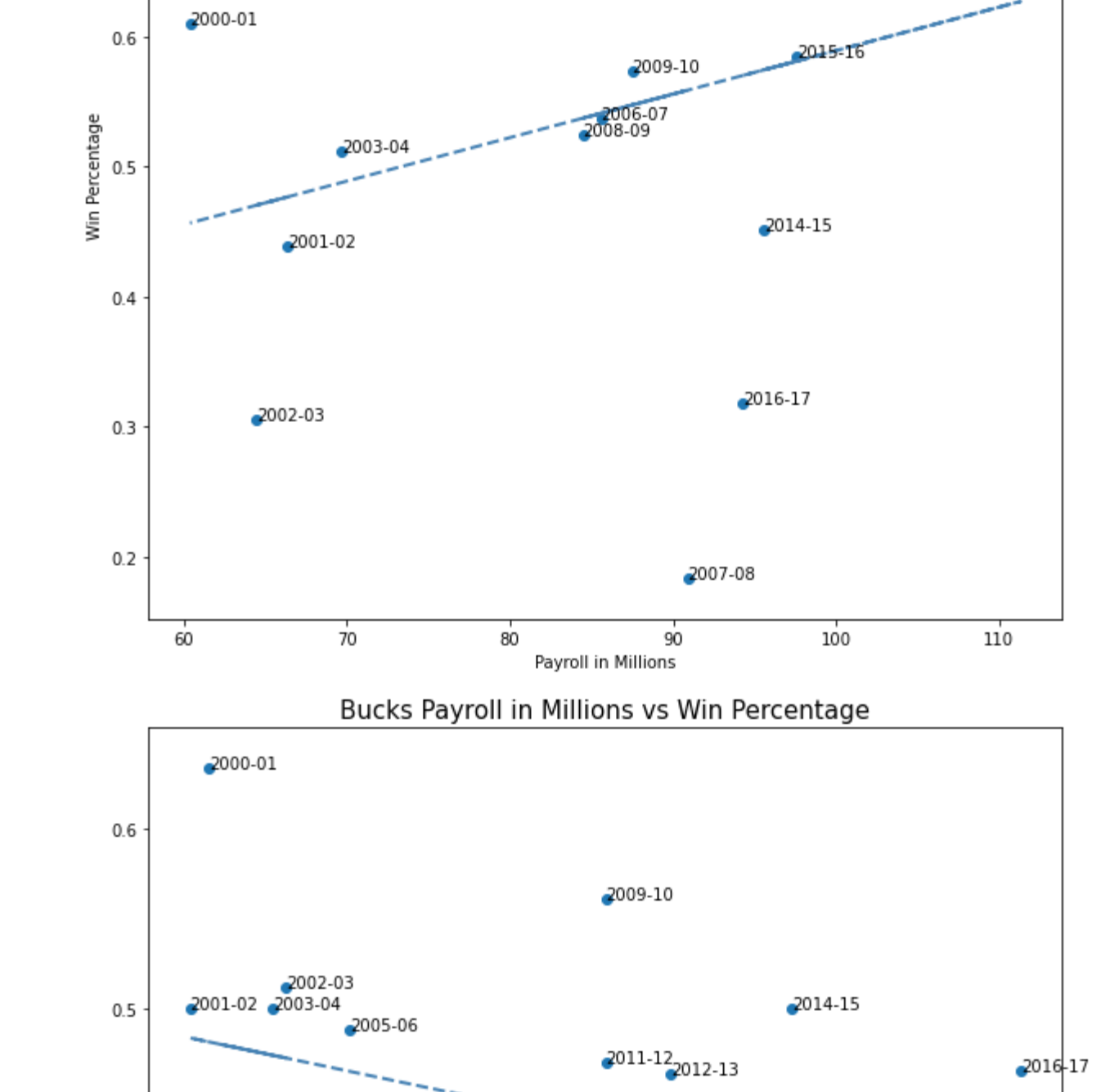


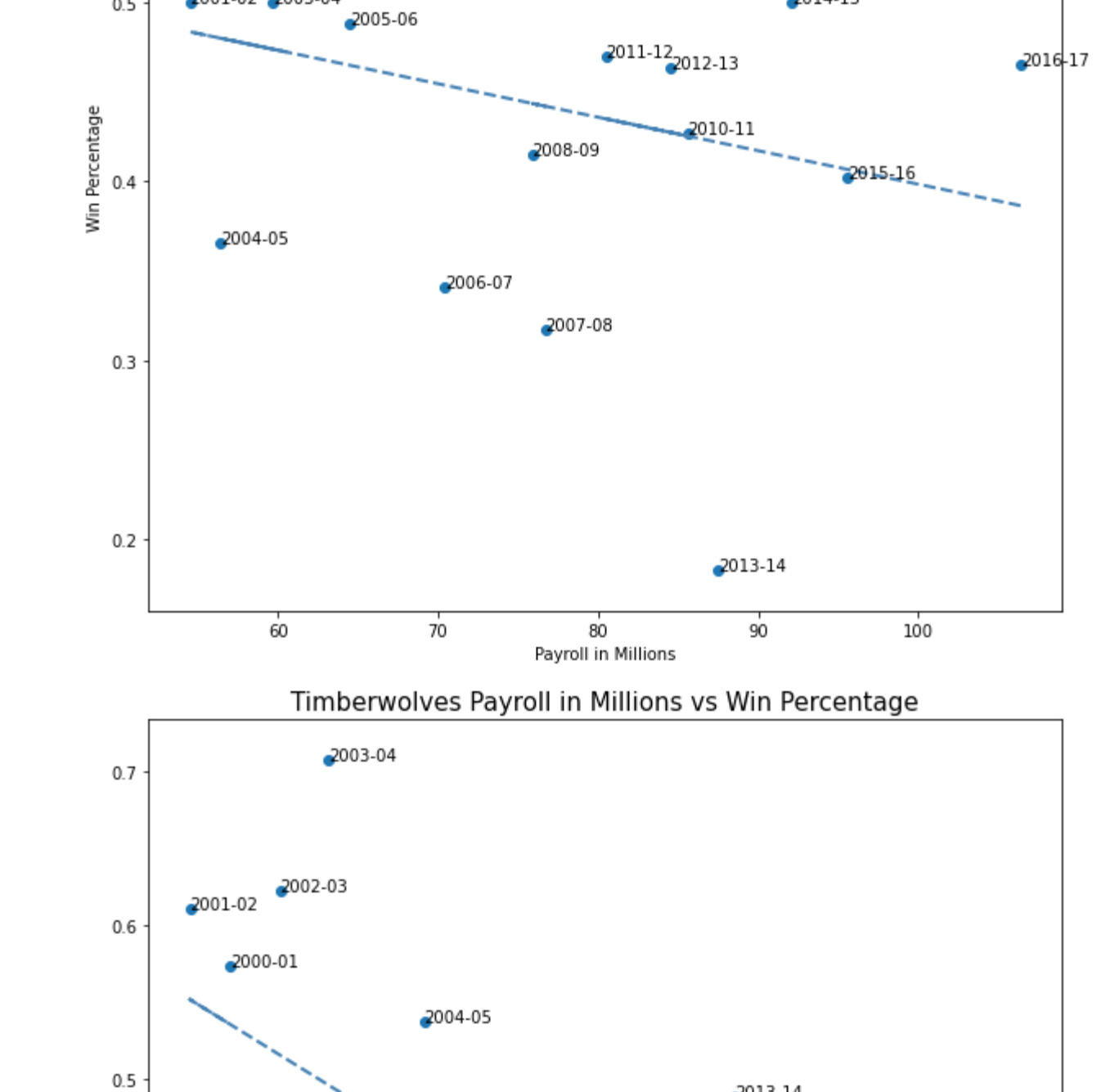


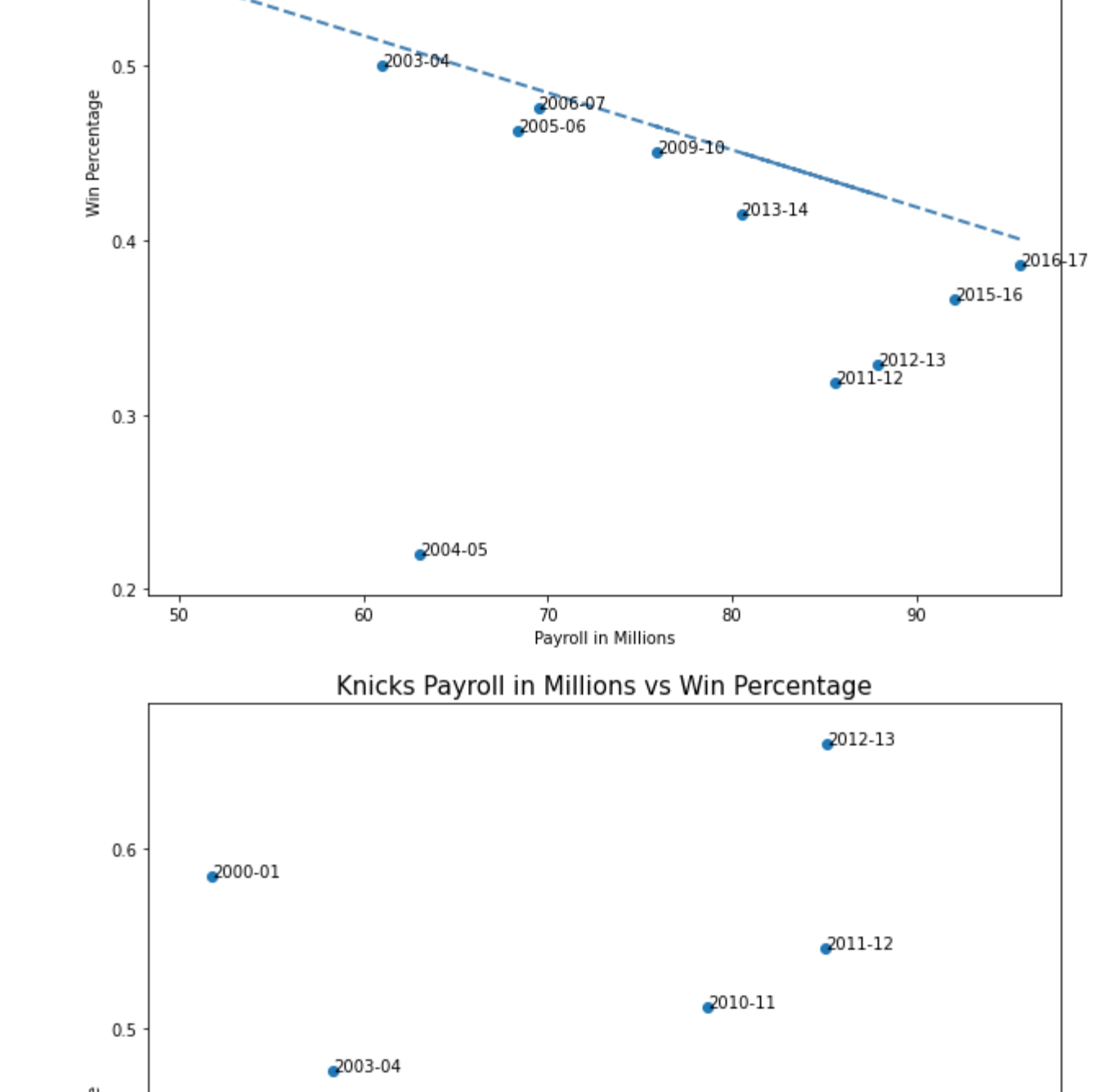


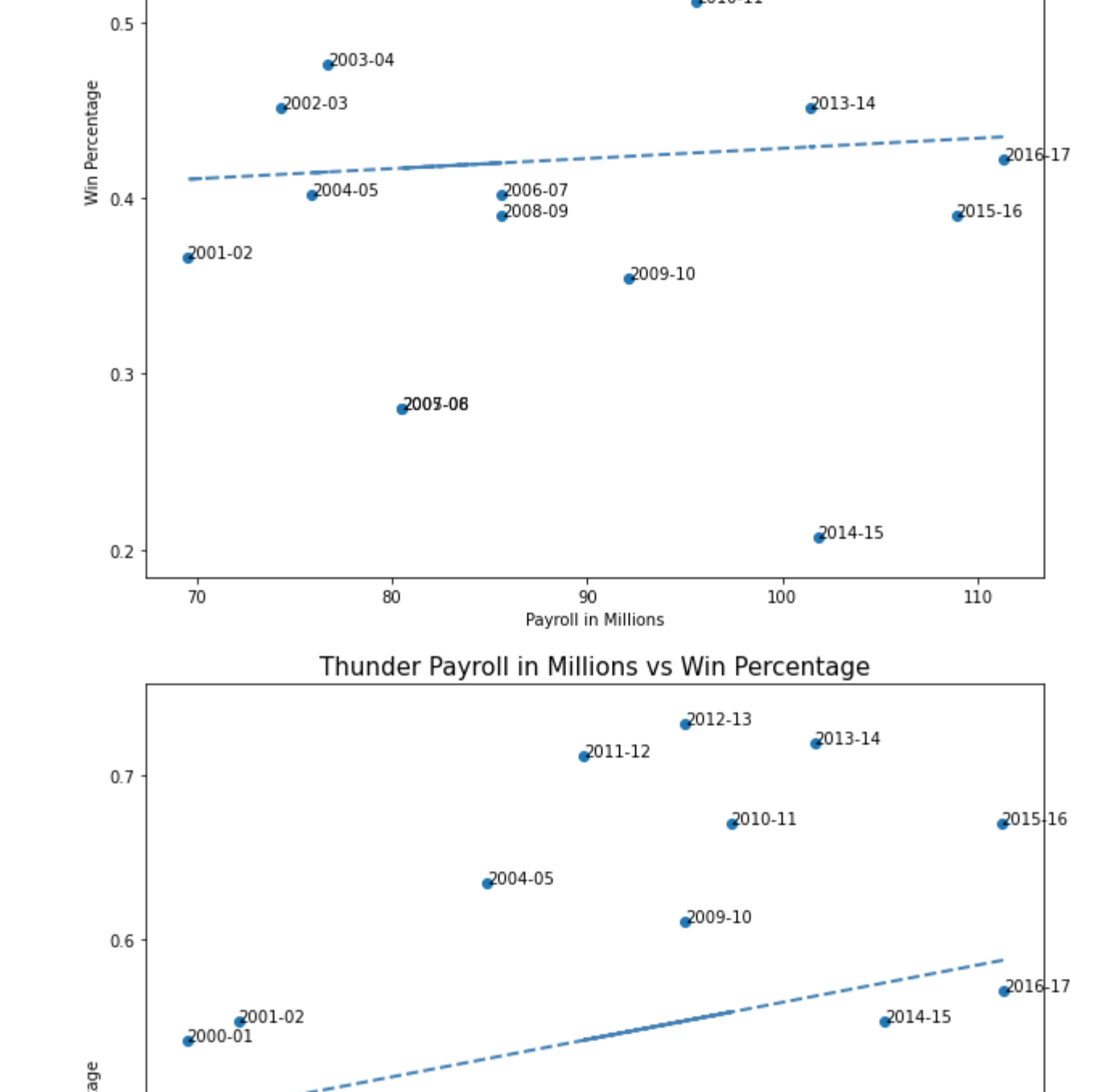


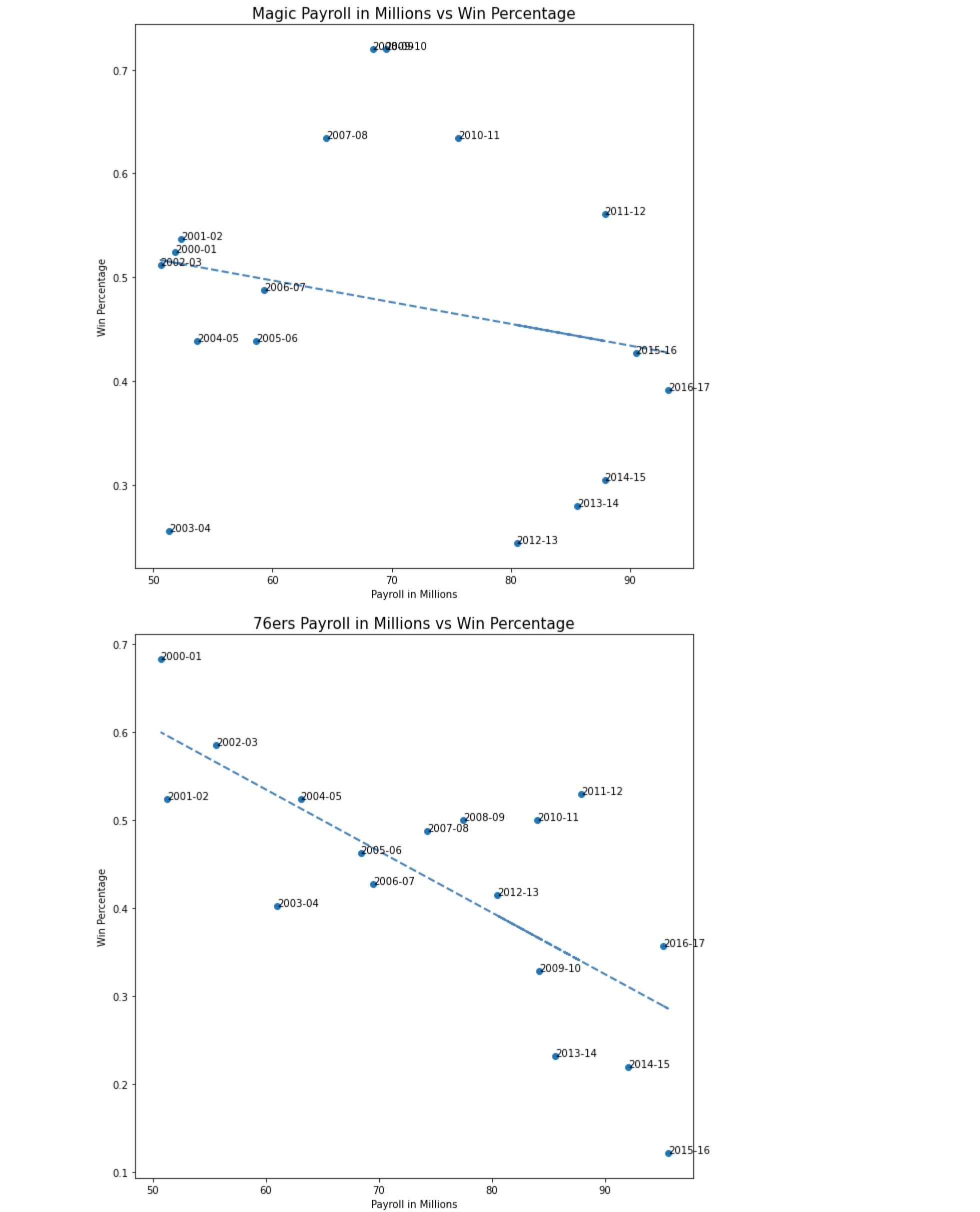


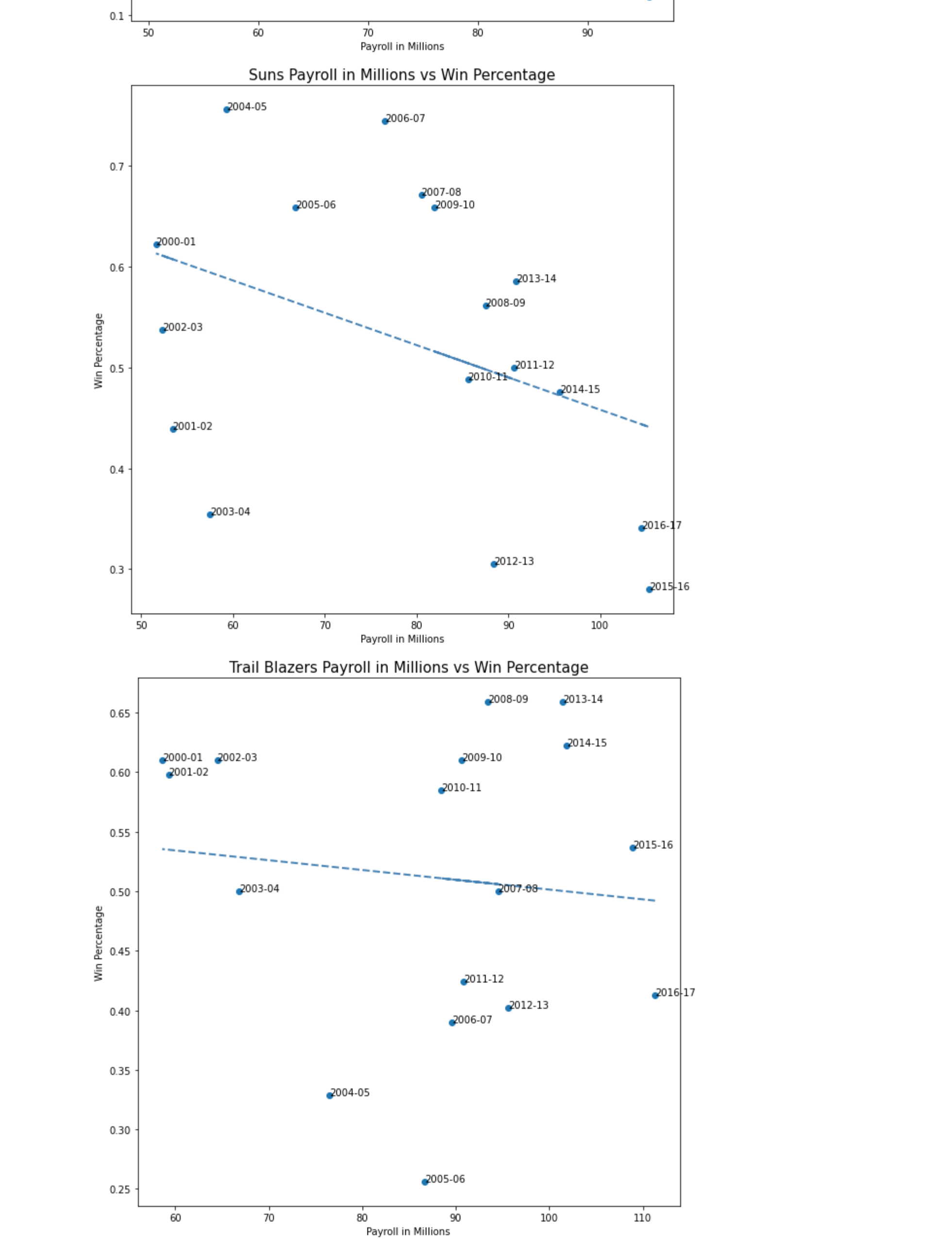


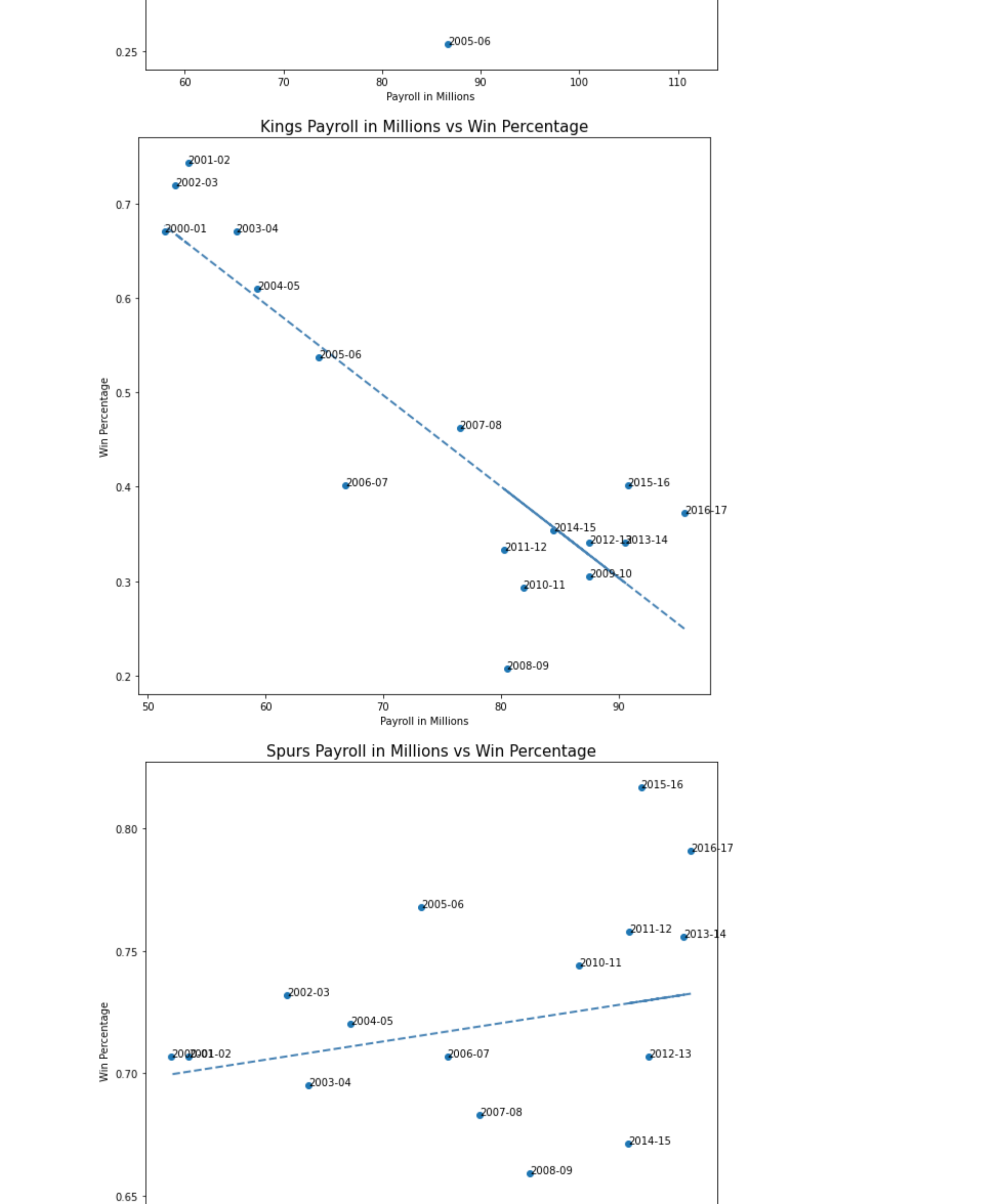


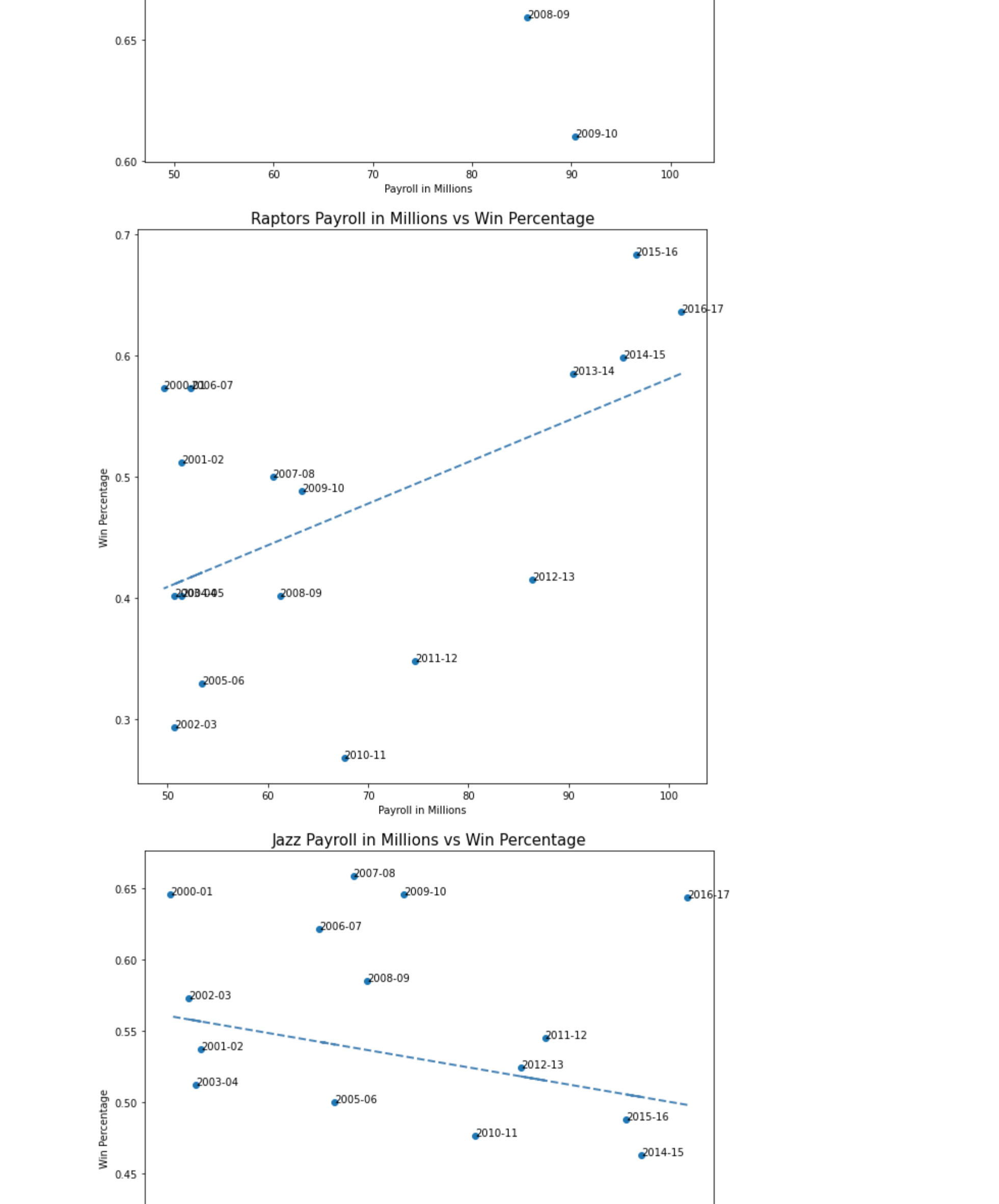


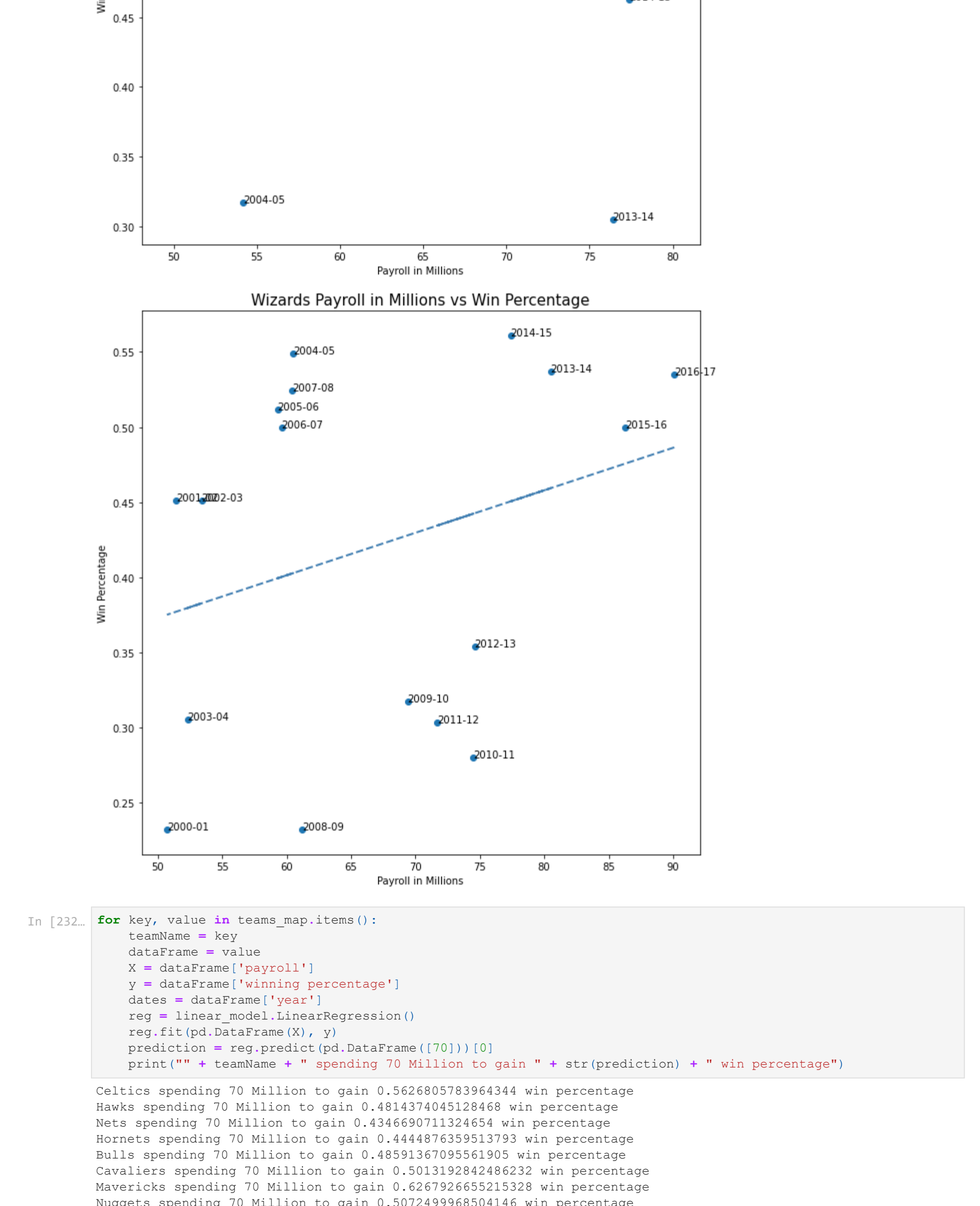


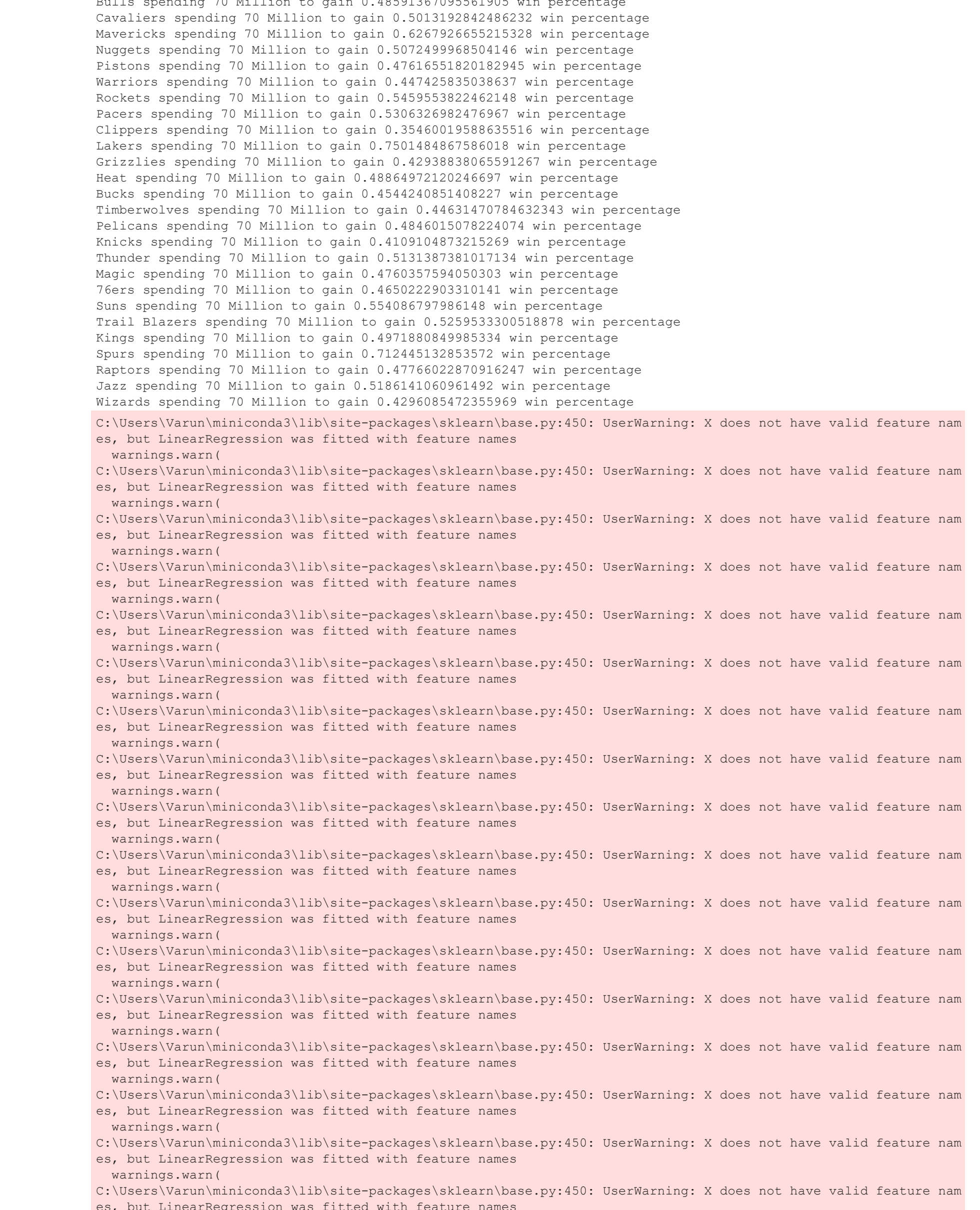












In [232…

```
for key, value in teams_map.items():
    teamName = key
    dataFrame = value
    X = dataFrame['payroll']
    y = dataFrame['winning percentage']
    dates = dataFrame['year']
    reg = linear_model.LinearRegression()
    reg.fit(pd.DataFrame(X), y)
    prediction = reg.predict(pd.DataFrame([70]))[0]
    print("" + teamName + " spending 70 Million to gain " + str(prediction) + " win percentage")
```

```
Celtics spending 70 Million to gain 0.5626805783964344 win percentage
Hawks spending 70 Million to gain 0.4814374045128468 win percentage
Nets spending 70 Million to gain 0.4346690711324654 win percentage
Hornets spending 70 Million to gain 0.4444876359513793 win percentage
Bulls spending 70 Million to gain 0.48591367095561905 win percentage
Cavaliers spending 70 Million to gain 0.5013192842486232 win percentage
Mavericks spending 70 Million to gain 0.6267926655215328 win percentage
Nuggets spending 70 Million to gain 0.5072499968504146 win percentage
Pistons spending 70 Million to gain 0.47616551820182945 win percentage
Warriors spending 70 Million to gain 0.447425835038637 win percentage
Rockets spending 70 Million to gain 0.5459553822462148 win percentage
Pacers spending 70 Million to gain 0.5306326982476967 win percentage
Clippers spending 70 Million to gain 0.35460019588635516 win percentage
Lakers spending 70 Million to gain 0.7501484867586018 win percentage
Grizzlies spending 70 Million to gain 0.42938838065591267 win percentage
Heat spending 70 Million to gain 0.48864972120246697 win percentage
Bucks spending 70 Million to gain 0.4544240851408227 win percentage
Timberwolves spending 70 Million to gain 0.44631470784632343 win percentage
Pelicans spending 70 Million to gain 0.4846015078224074 win percentage
Knicks spending 70 Million to gain 0.4109104873215269 win percentage
Thunder spending 70 Million to gain 0.5131387381017134 win percentage
Magic spending 70 Million to gain 0.4760357594050303 win percentage
76ers spending 70 Million to gain 0.4650222903310141 win percentage
Suns spending 70 Million to gain 0.554086797986148 win percentage
Trail Blazers spending 70 Million to gain 0.5259533300518878 win percentage
Kings spending 70 Million to gain 0.4971880849985334 win percentage
Spurs spending 70 Million to gain 0.712445132853572 win percentage
Raptors spending 70 Million to gain 0.47766022870916247 win percentage
Jazz spending 70 Million to gain 0.5186141060961492 win percentage
Wizards spending 70 Million to gain 0.4296085472355969 win percentage
```

```
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\Varun\miniconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature nam
es, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [ ]: