Jeffrey Lee-Mars
Data Science 2015
General Assembly

Introduction

The United States Bowling Congress runs a yearly 5 month long tournament, with participants numbering anywhere from 30,000 to 40,000. Bowlers can qualify for 1 of 2 divisions, in which qualifying is based on a yearly average that all bowlers must attain during the course of a bowling league.

The USBC provides a site that has scoring data for this year as well as previous years results. The focus of my project was to take the available data and see if I could glean any insights similar to the way that other major sports have used data in recent years.

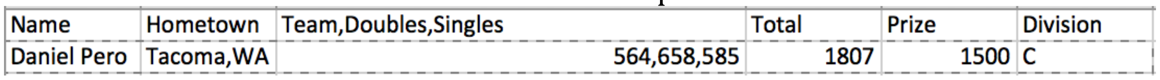
Due to the limit of the type of data available I had to ask very specific questions from the start. Can I possibly predict anything? Can I predict the likelihood someone will score well, or poorly in different squads.

As I worked through the data, and as you will see, I landed upon a question that is fairly prevalent in the bowling world. Can I use this data to possibly predict "Baggers". Baggers are bowlers who purposely bowl badly during their qualifying league to establish a low average, which would allow them to qualify for the lower division. Obviously the benefit of this would be for "better" bowlers to play in a division where the skill level isn't as high, and thus they would be more likely to make money. Using the data, could I identify questionable bowlers? And could I build a model to predict it.

Data Collection & Processing & Modeling

Using Kimono Labs, I was able to go to the USBC site and build a quick api to scrape scoring data for each squad as well as each division. The resulting apis allowed me to pull 6 CSVs per year, with a few hundred bowlers per file. From this data, I chose to focus on the CSV, which had the "All Events" data. The reason for choosing this data set was that it allowed for the most complete set of data, as it includes total score data from the other events. The one problem with this data was that it didn't include individual games within that total. I could have attempted to pull in the data to see how each bowler bowled by game. But I made the determination that all that mattered ultimately was the total for each squad.

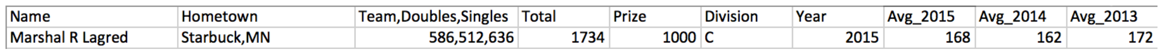
The initial data set included bowlers in this setup:



Name	Hometown	Team,Doubles,Singles	Total	Prize	Division
Daniel Pero	Tacoma,WA	564,658,585	1807	1500	C

Jeffrey Lee-Mars
Data Science 2015
General Assembly

After determining the goal of looking for people who could possibly be cheating, I had to go in and pull additional data that included bowlers averages for the last 3 years:



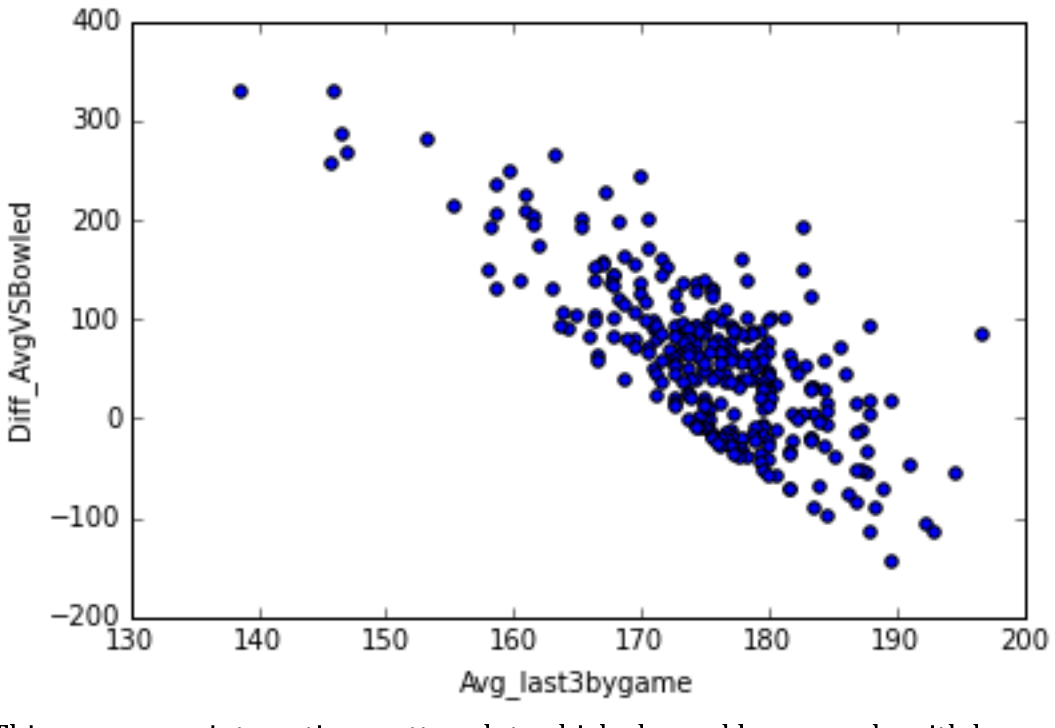
Name	Hometown	Team,Doubles,Singles	Total	Prize	Division	Year	Avg_2015	Avg_2014	Avg_2013
Marshal R Lagred	Starbuck,MN	586,512,636	1734	1000	C	2015	168	162	172

My first stab at analyzing the data was to do some feature engineering with bowling average and total pin counts. I created 2 new column: one that takes the mean from the averages of the last 3 years, the other taking the total pin count and subtracting the average in order to create a difference in the "excepted" result.

```
df['Avg_Total_2015'] = df['Avg_2015']*9
df['Avg_Total_2014'] = df['Avg_2014']*9
df['Avg_Total_2013'] = df['Avg_2013']*9

df['Total_Avg_last3'] = df.Avg_Total_2015+ df.Avg_Total_2014 + df.Avg_Total_2013
df['Avg_last3'] = df['Total_Avg_last3']/3

df['Diff_AvgVSBowled'] = df['Total'] - df['Avg_last3']
```



This was a very interesting scatter plot, which showed how people with lower averages tended to bowl much better then expected. But some negatives with this was that it didn't cluster very well, and it didn't take into account an aspect of "Baggers" that I know as someone who bowls, but that the computer has no concept of.

Jeffrey Lee-Mars
Data Science 2015
General Assembly

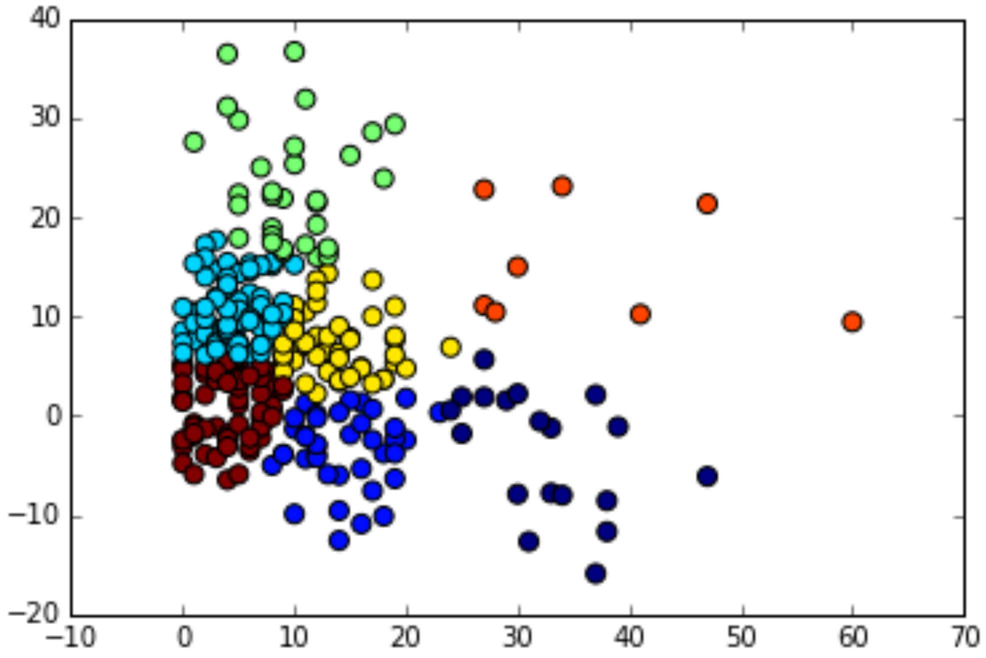
As a human, when looking at bowlers averages over the last few years, something that sticks out as suspicious is when a bowler had a higher average in previous years but then had a lower average in the current year. This would indicate that the bowler is of a higher skill level then his average would otherwise represent. The scatter plot above does not take this aspect into account. Bowlers with wild swings in average are not represented very well because the high averages pulls the low average to a higher number. To solve this, I had to do some additional feature engineering.

```
def avg_range(row):
data = [row['Avg_2013'], row['Avg_2014'], row['Avg_2015']]
return max(data) - min(data)

df['avg_range'] = df.apply(avg_range, axis=1)
df['avg_range_total'] = df['avg_range']*9
```

I created a new column that looked for the min and max of the 3 provided years of average a bowler, and took the difference. Effectively this created a variable that measures the difference between a bowlers average over the last few years.

I then took this new variable and compared it against the pins over expected variable and got a much more interesting cluster.



This chart basically compares the range of average difference vs the pins over expected. This helps to take into account how people bowled higher then expected, but also the amount of change their average has seen over the last 3 years.

The important clusters are the 2 groups that appear at the top of the graph. These are generally people that would should be flagged, though the ones on the right are more suspicious then the left.

Using these clusters, I decided to try to generalize the 7 clusters into 2. Basically to attempt to predict whether a bowler could be flagged as suspicious. With Logistic Regression, I get the following.

```
Logfeatures = ['Total_avgBG', 'avg_range_total', 'avg_range']
X = df[Logfeatures]
y = df.Log_Bagger_Response
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
zip(Logfeatures, logreg.coef_[0])
```

```
[('Total_avgBG', 0.21572826885896421), ('avg_range_total',
0.0070766234160809767), ('avg_range', 0.00078629149071082554)]
```

```
scores = cross_val_score(logreg, X, y, cv=5, scoring='accuracy')
array([ 1. , 0.98305085, 1. , 0.96551724, 0.98275862])
```

While this was a pretty accurate attempt, I wanted to also increase the number of outputs to increase the type of flagging. Keeping the large non flagged group, and splitting the flagged group into 2 groups, as “Baggers”, “May be Baggers”

To do this, I mapped the clusters to 3 different responses. And then ran KNN.

```
KNNfeatures = ['Total_avgBG','Avg_last3bygame','avg_range']
KnnX = df[KNNfeatures]
Knny = df.KNN_Bagger_Response

knn = KNeighborsClassifier(n_neighbors=5)
scores = cross_val_score(knn, KnnX, Knny, cv=5, scoring='accuracy')
np.mean(scores)

array([ 0.96610169, 0.96610169, 0.93220339, 0.98275862, 0.98275862])

0.96598480420806543
```

This also rendered a pretty accurate model that helped to split the flagged group into a more actionable set.

Jeffrey Lee-Mars
Data Science 2015
General Assembly

Conclusion

With this data, I didn't have a clear way to use it in regards to applying it to a machine learning model. But using Kmeans to cluster I was able to create an output and that could then be combine with supervised machine learning models. This has been a valuable lesson because it allowed me to take data that didn't have any obvious responses.

In terms of what I discovered in the data, using the limited data available, I was able to generate a model that would allow USBC officials to flag bowlers for individual investigation to determine if the offender was actually cheating.

Things that I would look to follow up on would be applying this model with more data from previous bowling years to see if the results still hold. Also I would investigate to see if there were more options to pull data. Or if there are other data sources that would allow me to make different discoveries.