

Abstract

In high school cross country, there are ways that coaches and student-athletes can get a glimpse of which teams will appear to have the best chance of victory in a meet. On one website, athletic.net, this is done by a virtual meet. In doing so, the website will compile a list of the season best times of athletes competing in a particular meet. This can be done for invitational meets, conference championships (created by the coaches themselves), or regional/state championships (provided for by athletic.net, in this case). The list is then scored like a regular cross country meet, and the results are presented for the viewer to analyze. While it does provide some insight, this is not the most accurate way to predict the results of a cross country meet. I take into consideration the chance that athletes perform better or worse than their season best times, and compile that alongside the results supplied by the website.

1. Introduction



My high school cross country coach always used athletic.net as his go-to hub for insight on the upcoming regional championships. During the 2016 season, athletic.net's hypothetical meet function predicted our team (Battle Creek Harper Creek HS) would win the regional meet that season, finishing a dozen or so points ahead of other schools like Otsego HS and Parma Western HS. We felt confident going in, as

we were all in great shape, having recently run personal bests in the two weeks or so prior to that all-important day.



Recently however, something dawned on me. Four years since my high school won its first regional championship in 36 years, only a dozen or so points is not a lot in cross country. If our team was not racing very well up to that point, we very well could've lost. As coaches will say: "every point counts."

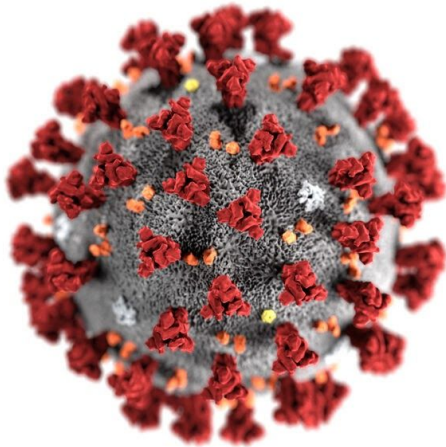
Cross country meets are scored based on the place an athlete finishes. If an athlete finishes first, that athlete scores one point. Second place scores two points, third place scores three points, and so on. Each team's top five runner's score's are part of a team's final score. The sixth and seventh runners for a team are still considered varsity, however their purpose is to finish ahead of other runner's top five finishers, in an effort to bump up other team's final scores. This is how my high school team was able to finish first place in the region as mentioned earlier.

Given this, athletic.net could not take into account how good or bad each athlete is feeling prior to the regional championship meet. As mentioned earlier, athletic.net only takes into account each athlete's personal best time for that season. Not how long ago the athlete ran that season's best time, for

example. So to refine the prediction process for a region, this program will aim to take into account each athlete's ability to perform differently compared to their recorded season best mark. Some athletes perform exceptionally well, while other athletes at the same meet might not. This variation (which athletic.net does not take into consideration) can make or break a team's chances to move onto the state championships.

2. Acknowledgement of Limitations

I'd like to preface the remainder of this report by noting one very notable obstacle in the way when testing this project: the 2020 Coronavirus (COVID-19) Pandemic.



Because of the pandemic, the Michigan High School Athletics Association (MHSAA) issued new qualifying standards for the cross country postseason, in an effort to maintain social distancing guidelines. Each region's teams are split into two groups, declared as sub-regions, consisting of about 8 teams. That sub-region would compete to qualify for the regional meet.

Traditionally, every team in the region would automatically go straight to the regional meet, with no need for a sub-regional qualifying round. Because of this, less teams would be running at the regional meet than in years prior, meaning any predictions generated by this program would to some degree be incorrect.

This code I am presenting also doesn't put the proper emphasis on the sixth and seventh runners. The algorithm I developed (and will go into more detail later) treats every runner as if they're scoring for their team. As I've previously gone over, sixth and seventh runners do not score. However, the algorithm does take into account one very important detail that does pertain to the sixth and seventh runners: their ability to still run fast and impact other team's scores.

3. Gathering the Data

Before the code is run, it's worth pointing out that this program writes a class named *Runner* to a new python file, called *dataset.py*. This is where the information we scrape will be stored, so it doesn't take up memory space when running our main file.

As mentioned previously, the data collected for this project is gathered from the hypothetical meet page on athletic.net. This list takes only into account each team's top seven athletes, no more. This is to prevent an eighth, ninth, or so runner for a team messing up predictions for the rest of the region. After all, only seven varsity athletes are allowed per team. This information is easy to scrape from the website using Python modules *BeautifulSoup*, *requests*, and *urllib.request*.

These modules when executed in a *for loop* will scrape through each line of data, and save the information to a string. Upon starting the program, the user will be prompted to insert the URL to the hypothetical meet that the user wishes to be analyzed into a window. When the user pastes the URL and closes the window, the program then scrapes the website, looking through every line of HTML in the website, saving information and writing it to the aforementioned *dataset* file. The aforementioned loop stops when the string *javascript* is detected in the HTML code. This stops the loop and moves onto the next section of the code.



It's worth including that in the *dataset* Python file, all of the information for each athlete is written in instances of the *Runner* class mentioned earlier. In that class' *__init__* function, the code is compiled into a tuple and saved to an array, which is then imported back into the main file as the variable *data*.

4. KMeans

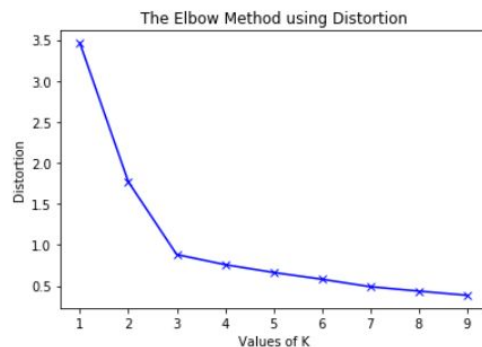
4.1. Introduction

Each of the values in the *data* array when imported has different variables that

will become important later on. Those variables are:

- Runner name
- Runner's school
- X-coordinate
- Y-coordinate
- Runner's season best time
- Season best time's date

The X and Y coordinates mentioned above are determined in the *Runner* class that was written in the *database* file earlier. These coordinates are set on a DataFrame (imported from the Pandas module) and evaluated through the KMeans clustering algorithm. Before we can sort our dataframe, we have to determine the number of clusters to sort the DataFrame into. This is done by the *elbow_method* function¹, which will run KMeans multiple times in an effort to find the most optimum number of clusters for our dataset.

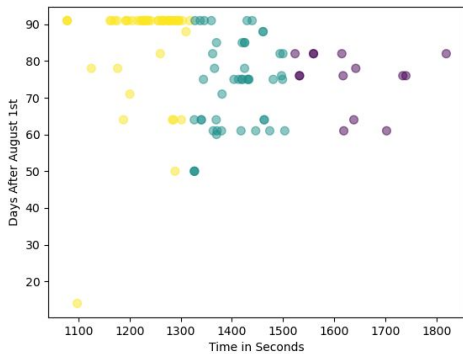


4.2. Sub Clustering

4.2.1. Analysis

The next step in our predictions is to permute each cluster of data. This is done

¹ The *elbow_method* function was developed by the user AlindGupta, and adapted to fit with the program developed.



by using the *permutations* class from the *itertools* Python library. Unfortunately when we attempt to run *permutations* on our clusters, we run into a *MemoryError*, because our list of permutations is still too large for Python to handle. The cluster assignments we just made are going to have to go through another round of clustering before moving onto the permutations. This is done by declaring more DataFrames and running the *elbow_method* function again to run the clustering process again. This is done repeatedly until we have subclusters that have about 10 athletes or less. Once we get to this point, we can move onto the permutations of each subcluster.

4.2.2. Theory

Doing multiple rounds of permutations also goes into one of the main ideas for this project in the first place. The idea is that any athlete has a chance to have a good or bad race on the day of an important race. This allows room for athletes to have the chance to move up if they're at the bottom of a cluster, but not too far. A runner who runs 19 minutes for a 5k consistently isn't going

to go run a 16:15 for a 5k in the span of two weeks, for example.

5. Permutations

5.1. Process

This is finally when we get to scoring each subcluster. As mentioned earlier, this is accomplished by using *permutations*. At this point, there are already some dictionaries and lists declared to store the results of each permutation's score. By using a several *for loops*, we can go through each possible permutation and score it accordingly by finding each athlete's school. These scores are saved to the dictionaries mentioned earlier. From this point, the average of each school's list is taken, and then sorted to achieve our clustered results.

5.2. Issues

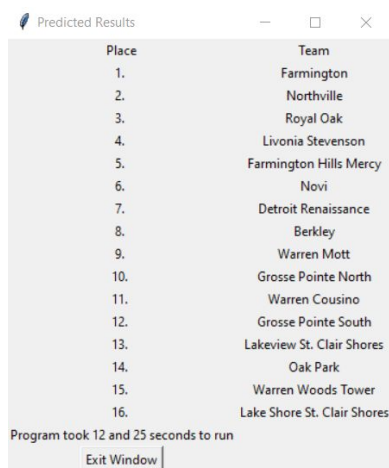
5.2.1. Timing

When this program is running, this is the section that takes the most time. In fact, it'll take around 12 minutes for this section to be completed. Many of the early clusters will have up to several hundred thousand different permutations. Given this, it's obvious why the program would take so long in this current form. Perhaps this could be addressed in future versions.

5.2.2. Clustered Results

When looking at the initial clustered list, we get a list that looks interesting. In some places, we get results that are similar to the athletic.net list earlier, while in some places the list looks completely different. In the

example here, comparing the athletic.net list to the clustered list we compiled shows the



Place	Team
1.	Farmington
2.	Northville
3.	Royal Oak
4.	Livonia Stevenson
5.	Farmington Hills Mercy
6.	Novi
7.	Detroit Renaissance
8.	Berkley
9.	Warren Mott
10.	Grosse Pointe North
11.	Warren Cousino
12.	Grosse Pointe South
13.	Lakeview St. Clair Shores
14.	Oak Park
15.	Warren Woods Tower
16.	Lake Shore St. Clair Shores

Program took 12 and 25 seconds to run

Exit Window

Warren Cousino HS women's team going from 15th place on athletic.net up several places on our list. There's always a chance that the ladies on that team could all have a great race, but jumping that many spots in cross country is quite a zany prediction to make.

6. The Final Stretch

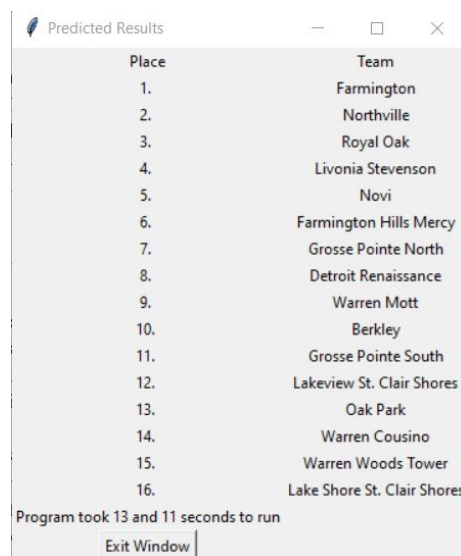
To avoid the issue of zany predictions like before, we'll take the average of the two lists: the list we compiled by clustering and the list from athletic.net. While the athletic.net list has its flaws we mentioned earlier, it still has some validity to it. Primarily because the list compiled on athletic.net doesn't make very zany predictions because its algorithm is much easier to compile than the list we made. Thus, we can in theory combine the two lists to make our predictions not so outlandish. This is done by taking the average place of each school in the two lists and sorting them accordingly. What we finally get after all of this is our final predictions at long last. All

that's left to do is display the results, which in this program is accomplished by creating a GUI window using *tkinter* and utilizing a *for loop* to add the results to the window. The total time to run the program is also added to the GUI window.

7. Discussion

7.1. Disclaimer

I've already gone over how the COVID-19 pandemic has impacted the qualification process for high school cross country. However the results that this program developed are compiled in a way that it fulfills its goal for the time being.



Place	Team
1.	Farmington
2.	Northville
3.	Royal Oak
4.	Livonia Stevenson
5.	Novi
6.	Farmington Hills Mercy
7.	Grosse Pointe North
8.	Detroit Renaissance
9.	Warren Mott
10.	Berkley
11.	Grosse Pointe South
12.	Lakeview St. Clair Shores
13.	Oak Park
14.	Warren Cousino
15.	Warren Woods Tower
16.	Lake Shore St. Clair Shores

Program took 13 and 11 seconds to run

Exit Window

7.2. Results

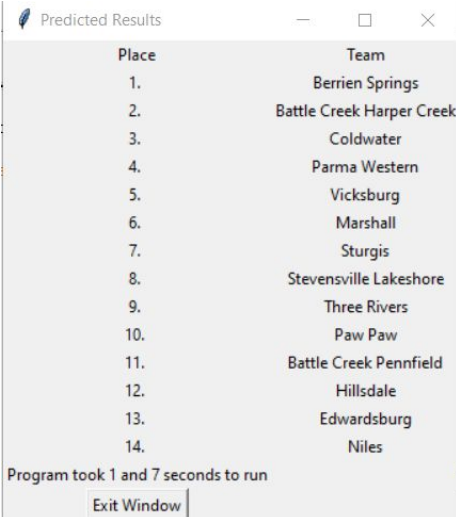
7.2.1. Division 1 - Region 8 (Women)

This region is the same region that we referenced earlier when we brought up Warren Cousino's women's team. These final results show that the top teams finish in the exact order that was reported on athletic.net for the regional meet in late

October. The two teams that were keys to the program's success were Livonia Stevenson and Novi. Novi's team had the edge according to the hypothetical meet on athletic.net, however when clustered and sorted Novi was placed behind Livonia Stevenson's squad. Both teams didn't qualify for the state meet in the end, but this is one sign of early success.

7.2.2. Division 2 Region 12 (Men)

Going to the region with my high school team, it's clear that the prediction machine developed still is in it's infant stages. The prediction made here was that Berrien Springs would secure a victory over Harper Creek, however that did not happen. In fact, several teams had much better races as a group than the machine could've predicted. Notably, Vicksburg HS finished in fifth place according to the predictions developed here, when in fact Vicksburg's team ran their hearts out, qualifying for the state meet in 3rd place, ahead of Stevensville Lakeshore and Parma Western.



Place	Team
1.	Berrien Springs
2.	Battle Creek Harper Creek
3.	Coldwater
4.	Parma Western
5.	Vicksburg
6.	Marshall
7.	Sturgis
8.	Stevensville Lakeshore
9.	Three Rivers
10.	Paw Paw
11.	Battle Creek Pennfield
12.	Hillsdale
13.	Edwardsburg
14.	Niles

Program took 1 and 7 seconds to run

Exit Window

7.3. Future Work

With this current iteration of the project, there's some more work that can be done to add more predictions to the program, as well as to make the program better at predicting results. One problem that could be troubling is when two schools are tied for a place. Early on, the tie breaker built into Python's *sorted* method settles ties by giving the edge based on alphabetical order. This will be a quick fix in the future, but at the time of this report, there's not enough time to implement that into the program.

As in the case of Vicksburg's better-than-expected performance in their regional meet, I expect a future iteration of this program to run through the clustering several more times, as to gather a wide variety of possible outcomes for a particular meet. As it stands, this program only runs through one pass through the clustering and sub-clustering process. And while that successfully predicted the finishing results of Novi and Livonia Stevenson, it failed to successfully predict the results for Vicksburg.

One last feature that needs further addressing: better implementation of web scraping. Personally a newer topic that I had to learn on my own, assigning date values early in the code that were gathered from web scraping is running into a lot of errors when creating the dataset.

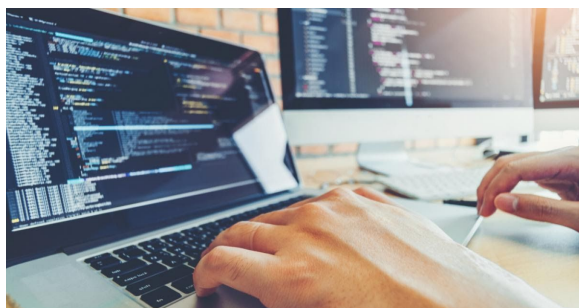
8. Final Thoughts

I believe I bit off more than I could chew with this project, considering how much time in the semester we had to work on it. Granted during the course of this class, we had to go from complete novices in the realm of machine learning to developing our own machine learning projects. Can this project be improved on? Of course, but it will take more time to do so outside of the (virtual) classroom.

The points I mentioned in the Future Work section of this report will be what I will work on in the future months in an effort to make this program more accurate. All updates to this program will be saved on Github under the *Version 2* folder in the repository link below:

<https://github.com/Qflat/Cross-Country-Predictor>

The repositories are also going to be open to the public, so that any other developers can look at my work and make contributions if necessary in the future. But for the purposes of this project as it pertains to the class, this is the final product.



References

AlindGupta. "Elbow Method for Optimal Value of k in KMeans." *GeeksforGeeks*, 6 June 2019, www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/.

"Example of K-Means Clustering in Python." *Data to Fish*, 26 Mar. 2020, datatofish.com/k-means-clustering-python/.

Julia-Git. "Julia-Git/webcraping_ny_mta." *GitHub*, 24 Jan. 2020, github.com/julia-git/webcraping_ny_mta.

Photos/Other Figures

Boyd, Blake. *Ohio on Verge of Statewide Lockdown as COVID-19 Cases Rise*. 13 Nov. 2020, oxfordobserver.org/3371/health-safety/ohio-on-verge-of-statewide-lockdown-as-covid-19-cases-rise/.

Cross Country: COVID-19 Fall Guidance. 13 Aug. 2020, www.iahsaa.org/covid-19-fall-cross-country.

File:Python-Logo-Notext.svg. commons.wikimedia.org/wiki/File:Python-Logo-Notext.svg.

Heinz, Kate. "Software Engineer vs. Programmer: What's the Difference?" *Built In*, 29 Jan. 2020, builtin.com/recruiting/software-engineer-vs-programmer.

Track & Field and Cross Country Statistics. www.athletic.net/.