Framingham School bus Project

Yujia Wang, Yinzhu Su, Sang Jae Park

Note: For more detailed approach design, please refer to: Main approach draft and Detailed bus assign

1. Project Task

Recently Framingham Public Schools find that their costs and budgets are getting tighter, so they are going to figure out some ways to reduce their costs. The main idea is to cut the cost from their transportation department, more specifically, the number of school bus. So the goal of this project is to find an efficient way to reform the bus routes for minimize the quantity of busses

2. Data Set

18-19 School bus

3. Analyze the data

The original data has so many columns, which are as shown below:

```
Index(['specialed_1=door to door _ 0 = general ',
       'elg_code__ 0 = eligible _ 13 = ineligible', 'distsch___from school',
       'school____', 'grade_____', 'res_addr__', 'am_stptime', 'am_busnumb',
       'am_locatio', 'am_stop___', 'am_stpdesc', 'pm_stptime', 'pm_busnumb',
       'pm_locatio', 'pm_stop___', 'pm_stpdesc', 'am_dststop', 'pm_dststop'],
      dtype='object')
```

Actually we only need some of these:

```
Index(['school____', 'res_addr__', 'am_stptime', 'am_busnumb', 'am_locatio',
       'am_stop___', 'am_stpdesc', 'pm_stptime', 'pm_busnumb', 'pm_locatio',
       'pm_stop___', 'pm_stpdesc'],
      dtype='object')
```

And we renamed them to:

```
Index(['school', 'res_addr', 'am_stptime', 'am_busnumb', 'am_locatio',
       'am_stop', 'am_stpdesc', 'pm_stptime', 'pm_busnumb', 'pm_locatio',
       'pm_stop', 'pm_stpdesc'],
      dtype='object')
```

Note that the am_stpdesc and pm_stpdesc mean the stops the students live closest when in the morning and afternoon/evening.

And the total bus number now using is 74, it includes some big bus, small bus, and mini bus. And the school open time as well as some other information will be found here:Framingham school department hours

The total rows of the data is more than 9000, actually some rows(students) are just hidden because some overbooking problem. So the rows which have useful information is 6456, and we need to handle with these data.

The total number of school is 17, which are:

```
array(['ALT', 'BAR', 'BRO', 'CAM', 'CHA', 'DUN', 'FHS', 'FUL', 'HEM',
       'JUN', 'KNG', 'MCC', 'POT', 'STA', 'STB', 'WAL', 'WIL'],
      dtype=object)
```

So the final dataframe will be like:

Out[567]:

	school	res_addr	am_stptime	am_busnumb	am_locatio	am_stop	am_stpdesc	pm_stptime	pm_busnumb	pm_locatio	pm_stop	pm_stpdesc
2007	FHS	250 BALDWIN AV	06:39 AM	31	NaN	FHS.026	BALDWIN AV & HIRAM RD	02:06 PM	31	NaN	FHS.026	BALDWIN AV & HIRAM RD
2008	FHS	89 BETHANY RD	06:36 AM	28	NaN	FHS.117	WINTHROP ST & WAVERLY ST	02:16 PM	28	NaN	FHS.117	WINTHROP ST & WAVERLY ST
2009	FHS	89 BETHANY RD	06:36 AM	28	NaN	FHS.117	WINTHROP ST & WAVERLY ST	02:16 PM	28	NaN	FHS.117	WINTHROP ST & WAVERLY ST

4. Approach

Our main idea is described as the picture below. In this picture, we use the big orange spot in the middle to represent one school, and use the small blue spots around the orange one to represent all the stops we need to stop by for this school.

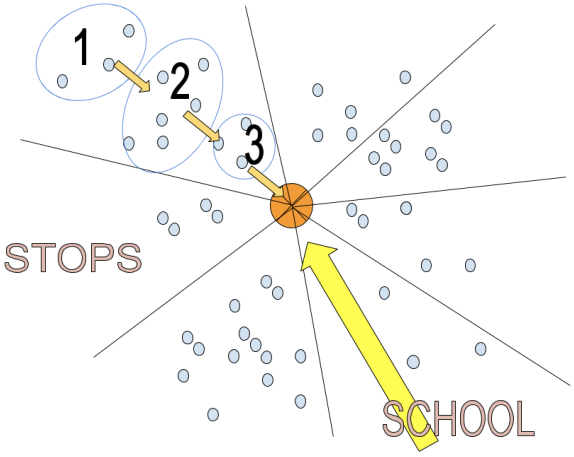
And we use google map API to get the coordinates of the stops. And then we calculate the angles between school and its stops.

Once we have the angels, we use K-means to cluster all of the stops only by their angles. In the picture below, we use line to divide the stops to make it like a pizza. For each cluster, we use one bus to go from the farest stop to the nearest stop between school.

And then we use left-top cluster as an example. For the left-top cluster, we divided them into three smaller clusters, only by their proximity to each others. In the picture below, we use three circles to represent the three smaller clusters. And then for each smaller cluster(circle), we calculate the mean distance between the school for the stops in the small cluster(circle). And we can sort the three means of distance.

Finally, we go from the farest mean-distance circle to the nearest mean-distance circle. And for each stop in the circle we only calculate their distance to the school, and sort them also from farthest to nearest. We also connect those blue spots by the sorted sequence so that it looks like a real road navigation.

Buy doing this method above, we can get one bus route to one school. The rest we need to is just iterate the method on other pieces of the pizza and also iterate on other schools.



5. Techniques

5.1 Python & Jupyter Notebook

We use Python and Jupyter Notebook for our testing. Python is a very useful programming language for data analysis. And Jupyter Notebook is a really good way to show some results and debugging.

5.3 Sklearn and Pandas

We also use python package sklearn and pandas for the data analysis.

5.2 Google Map API

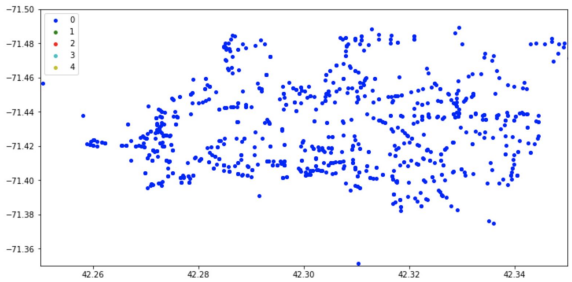
We use google map API to get the distance of each stop to the school, as well as the driving time of each two stops. And finally get the timeline of all the buses!

5.3 K-means

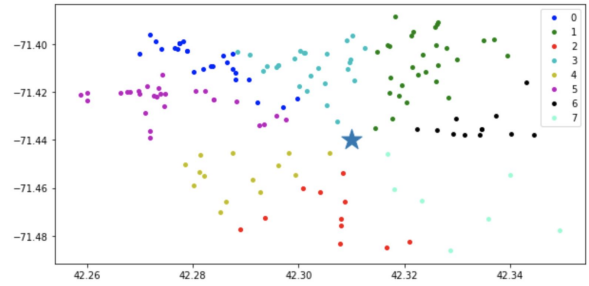
K-means is actually an unsupervised machine learning method. For the first 'pizza' clustering, we used the angles as the features to be input to the model, and we can easily cluster the stops like a 'pizza'. And then for each 'pizza', we used the K-means again to cluster the 'pizza' in three smaller clusters only by their proximity.

6. Visualization of main steps

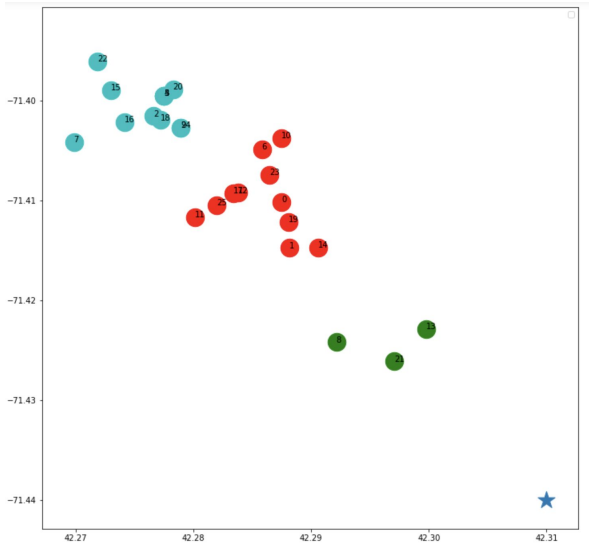
6.1 Stops visualization



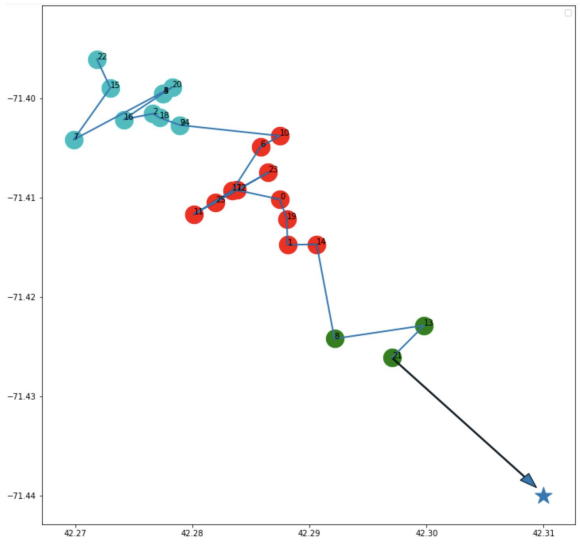
6.2 Divide them like a pizza (example for only one school)



6.3 Make an example for one slice of 'pizza'



6.4 Route the bus in this slice!



So far we made a rerouting example for one slice of stops, and next we implemented this algorithm for all the schools and buses.

6.5 Re-arrange the schedules

Since we have changed almost all the routes, after we rerouted for all stops we need to specify new schedule for each stop. We used Google Map API to generate new schedule. Google Map API can give us the driving time between two coordinates. Since we have all stops' coordinates and each school's Drop Off time, we can reversely calculate the time when each bus should arrive at each stop. Same thing for the new schedule of sending students back to their home.

And then there will be our results. Before we show our final result, we need to handle with some detailed problems first. The problems will be shown in next part.

7. Problem we found and how do we solve

Until now, we have get new routes and new schedules. However, we haven't arrange buses for each stops. This is a complex problem, because each route has different amount of students, and we also have three kinds of buses that has different capacity(and even different functions). Although most routes only need one bus to pick up all the students in the route to their school, there are still some routes that has too many students to be contained in one bus(even if the largest type). The easiest method to solve this problem is arrange more buses for this kind of routes. For example, if a route has 100 students, and the largest capacity of our buses is 70, then we have to send two 70-peoples buses to this route, or one 70-peoples buses and one 32-peoples buses. However, we only have limit amount of buses in total, if we arrange the buses like this, then the amount of buses we need will be much more than the original amount. Our goal is to save the total amount of buses to reduce budget, so we have to find better solutions for this problem. Then we were thinking of reusing the buses. We found that the Drop Off time for all the 17 schools in the system can be clustered into three main time period: high schools are earlier to have the first class, so their Drop Off time are around 7am; middle school are about one hour later, the Drop Off time are around 8am; elementary schools' students only need to arrive their school after 9am; and there is one kindergarten that has a Drop Off time of 10am. So we can let the buses that have sent the high school students to go back to the start points of middle school students, and after they have drop off all middle school students, they can drive back and send elementary school students and the kindergarten students.

And another problem is also about the school buses. Some school buses(M00 to M08) are wheelchair accessible. We think some routes may need these buses when there are students who need to use wheelchair. However, we haven't gotten any information about students' wheelchair demands, right now we just use this special buses as normal buses.

8. Results

We ran our method for all the 17 schools, then we get all the routes we need. Then we calculated the driving time by using Google Map API so that we can get the schedule for each stop and each school. Last but not least, we re-arranged the school buses for those routes. And finally, we found that we only need 67 school buses for both picking up all students to their schools and sending them back home, which means we saved 7 school buses from the initial 74.

9. Future Improvements

As we said in part7, there are some routes that have students more than the capacity of one largest bus. Our current solution is using two buses in one route, from the first stop till the destination, just like merge the two buses as one super-large bus. But sometimes it's a waste of resource. For example, we send two buses from the very beginning till the end, but in the early part of route, there are only a few students, the two buses are just driving the same route together with lots of empty seats.

We have thought about an optimization method: For a route that contains students more than the capacity of the largest bus, we still use one bus start from the beginning stop, driving the same route to school. And when this bus is fully loaded, it won't stop at later stops but directly drive to school. At the same time, another bus will start from that stop, picking up all rest students till school. Since we have the data about how many students are in each stop, we can calculate which stop should be the 1st bus fully loaded, and this will be the same stop for the 2nd bus to start.

And if most of students in this kind of busy route are quite close to school, we can even just use one bus: this bus will also start from the beginning stop, driving the same route to school like the original method. And when it is fully loaded, the bus must have been quite close to school, because we have assumed that most of students are close to school in this situation. Then the bus will also directly drive to school, without stopping at any later stops. And after this bus arrive school and drop off all the students on the bus, it will go back to the stop where it became fully loaded. This progress will be fast, because as we said before, this

"fully loaded stop" will be quite close to school. And this bus will run the rest stops normally, until school.
These optimizations can not only help us save more buses, but will somehow save the total driving time for a route. Since we don't have enough time for achieving this function before the deadline, we put it into future improvement.

10. Timeline and Roles

Note, each teammate should be assigned some non-trivial coding task.

Task	**Deadline**	**Lead**
Data cleaning	11/13	Yujia Wang
Google API testing	11/17	Sang Jae Park
Design and implement the main algorithm	11/27	Yujia Wang & Yinzhu Su
Implement get_driving_time algorithm and testing	12/03	Yinzhu Su
Final testing and detailed problem handling	12/07	Yinzhu Su & Yujia Wang
Prepare report and presentation	12/10	all