Report Card Generator

Authors: Aditi Nair (asn264) and Akash Shah (ass502)

Final Project for DSGA-1007, Fall 2015

Motivation

When we found several datasets on NYC public schools' performance published by the NYC Department of Education on the NYC OpenData website, we wondered to what extent the quality of public school education varied across the city. We decided to create a tool that would allow users to search individual public high schools in New York City by name and would output relevant performance metrics and visualizations. Since we were interested in comparing high school performance across neighborhoods, we extended our tool so that we could generate summary reports given a user-provided starting location and radius. Finally, we also created a tool by which we could generate citywide Top 10 rankings and reports from a user-generated school ranking system. These three related functionalities are found in the Names, Location, and Top 10 modes of our program.

Data Sources

We merged several datasets, all from the NYC OpenData website. They are:

- Department of Education High School Directory (2014-2015): https://data.cityofnewyork.us/Education/DOE-High-School-Directory-2014-2015/n3p6-zve2
- SAT Results (2012): https://data.cityofnewyork.us/Education/SAT-Results/f9bf-2cp4
- Department of Education School Performance Directory (2014-2015): https://data.cityofnewyork.us/Education/DOE-High-School-Performance-Directory-2014-2015/xahu-rkwn
- Department of Education School Performance Directory (2013-2014): https://data.cityofnewyork.us/Education/DOE-High-School-Performance-Directory-2013-2014/42et-jh9v
- Regents-based Outcomes (Class of 2010): https://data.cityofnewyork.us/Education/Graduation-Outcomes-Class-Of-2010-Regents-based-Ma/k8hv-56d7

Installation

In order to run our program you will need GeoPy (https://pypi.python.org/pypi/geopy).
Geopy can be installed easily using the pip command <pip install geopy>.

You will also need the open source version of ReportLab. Download instructions can be found at: https://bitbucket.org/rptlab/reportlab. Please consult the specific instructions for your machine type.

You can find the source files for our project at https://github.com/asn264/ReportCardGenerator. It will also be available in the Final Project repository on the course Github page: https://github.com/ds-ga-1007/final_project.

Running the Program

To run the program, enter <python report_cards.py> at the command line.
You will be prompted to choose one of three modes: ‘name’, ‘location’, or ‘top10’.

In ‘name’ mode you can provide the names of NYC public high schools and we will directly generate a report summarizing the performance of those schools. (Consult school_directory.csv to see the schools in our database.)

In ‘location’ mode you can provide an address or set of coordinates in one of the cities in our database and a radius. (Consult school_directory.csv to see which cities are valid.) You will have a choice of how many of the schools within the radius of your address to include in your report.

In ‘top 10’ mode, you can choose metrics from the various school performance metrics in our database, along with a weight for each metric reflecting how important that metric is to you. A score will be calculated for each school based on those metrics and weights, and a custom report will be generated for the 10 schools with the highest scores.

Reports can be found in PDF format in the directory ReportCardGenerator/reports. This directory also contains sample reports that were generated using each of the available modes.

Unit-testing

Unit-tests are located in ReportCardGenerator/unittests.
They can all be run from the main ReportCardGenerator directory with the command <python -m unittest discover unittests>.

Overview of Program Structure

There is a directory in ReportCardGenerator called data which contains a raw_data directory, a data.py file and a database.csv file. The raw_data directory contains the raw datasets described above. The file data.py processes, cleans, and formats the raw data and saves it in a single database as the database.csv file. The data.py file was run once before, and does not run at each iteration of the program. This is done so that the program runs faster and to limit the number of calls to the GoogleV3 Geocoder API (data.py makes a large number of calls to this service).

The utilities.py file contains a function called ‘load_session’ which loads the database.csv file described above into a dataframe, creates a dataframe containing the names of all the schools in our database, and

also creates a list of valid features for generating the custom metric in top 10 mode and for creating the content of PDF reports. These are loaded into our main file report_cards.py by importing utilities.py

The modules mode.py and filename.py contain functions that help obtain the user's choice of mode and filename, respectively. The user input is validated until a valid input is received, and then returned to the main program.

The modules names_toolkit.py, location_toolkit.py, and top10_toolkit.py each contain a class which represents an iteration of entering that mode. Each class contains functions that obtain and validate the user input, and ultimately use this input to retrieve the appropriate schools from the database. These schools are directly specified in names mode, are within the provided radius of the input address in location mode, or are the top ten schools with the highest score according to the features and weights specified by the user in top 10 mode. Before being returned to the main program in report_cards.py, each school is instantiated as a School object.

The school.py module contains the School class, an instance of which represents a school in our database, and a corresponding custom exception. The School class is used for input validation and contains the functionality to query the database for information that will be in our final PDF report.

The module summary_writer.py contains the SummaryWriter class, which generates the text content of the report and also integrates the relevant visualizations from the GraphGenerator class (below) into the report. The SummaryWriter class was built using the open-source ReportLab package and relies heavily on its SimpleDocTemplate class which provides a pre-formatted template for report-generation. Generally, the class collects relevant information - cast as Paragraph or Image objects in ReportLab - and the necessary formatting tools - mostly ReportLab's Spacer and PageBreak objects - into a large list and uses ReportLab's build function to generate the final PDF document.

The module graph_generator.py contains a class responsible for generating all of the plots used to visualize the data queried by user input. There are a variety of plots generated using matplotlib - box plots, histograms, and bar plots - although some may not always be displayed in a report based on the number of schools used to generate the report and the sparsity of the data in some categories. Once the visualizations are generated in png format, different graphing functions return the relevant file addresses, which are then used to integrate the visualizations into the final report using the SummaryWriter class.