# Predicting yields in the Univeristy of California system

## Abstract

We predicted the yield rate for California high school students matriculating to University of California campuses using a multivariate linear regression model. Our model used various descriptors of a high school (including GPA, SAT and AP scores, distance to campus, application and acceptance rates, and school size) to predict the yield rate for that high school in a given year to a given campus. We compared our results to a baseline model which predicted that the yield rate for a given high school would be the simply the average of the yield rates in the past. The linear regressor (error of 6.8%) performed better than our baseline model (error of 9.7).

## Introduction

In the realm of university admissions, the yield rate is the ratio of students who enroll at a university to the number of students who were accepted. Predicting yield rates is an important task for universities as they decide which students to accept into their programs. If universities underpredict the yield rate, then they will be overwhelmed with too many students who accepted their offers and enrolled. If they overpredict the yield rate, then not enough students will attend the school, and the university may face budget limitations due to tuition shortages. If the university could better predict the yield rate, then they could adjust their admission rates to better hit their target enrollment numbers. In this project we attempted to predict yield rates for University of California campuses from various California high schools.

Predicting yield rates is a very interesting problem because of the many different factors that could influence students' matriculation decisions. Among the many factors involved could be the desirability of the university, how far away the university is, the qualities of the student's high school, the qualities of the student themselves, and many other factors. Because of privacy concerns, we were not able to find any data on individual students, and thus had to be content with data aggregated over an entire high school. With this data we were able to make proxies for the desirability of the university, how far away the university is, and the qualities of the student's high school.

## Data Acquisition

There were three sources of data which we used. First, we found a dataset which included the number of applied, admitted, and enrolled students, as well as the average GPAs for those 3 groups, for a certain race of students from a certain high school to a certain college in a given year. This data came from University of California database located at https://www.universityofcalifornia.edu/infocenter/admissions-source-school. Then, we found another database which included SAT and AP scores, as well as student body size data, for a given high school and year. This data came from the California Department of Education at https://www.cde.ca.gov/ds/sp/ai/. Finally, we computed the distances from each high school to each UC campus using the Google Maps API. We were able to search for the distance between two places using a string description, such as “University of California, Berkeley” to “Abraham Lincoln High School, Los Angeles, California, USA”.

## Data Pre-Processing

There were many steps which we had to perform to clean, convert, and aggregate our distance datasets. This was by far the largest portion of our work, and probably where we learned the most.

First of all, for the GPA and count data, we had to rename columns, delete columns that were of no use to us, and split one column containing both name and school ID number into two different columns. Then, since the GPA and count data came in two different .csv files, we had to properly merge them together into one dataframe, so that one row contained both GPA and count data.

After merging these two dataframes together, we performed some additional cleaning steps such as deleting rows which contained all NaN values in the desired columns, splitting up two different location columns into four cleaner columns for city, county, state, and country to allow for easier filtering. Finally, we had to convert the dataframe so that one row contained all the information for one high school, UC campus, and year. The original data contained a different row for every measurement of applied, admitted, and enrolled, so we had to combine these sets of three rows into one row. After this preprocessing we saved the result so we could access it later quickly.

The dataset containing SAT and AP scores and total 12th grade class size was even harder to parse. It consisted of multiple `.xls` files grouped by year and test type. The `.xls` files had different formats. Examples of differences included extra/missing columns, the number of rows before the header, and different column names. This required us to parse each file on an almost individual basis. Merging all the data together resulted in two dataframes, one for SAT scores and the other for AP scores. The SAT score dataframe contained the average SAT score for each subject, as well as the number of students taking SATs, for each high school and year. We had to impute data in the SAT score section. That was necessary since in 2008 the SAT tests changed their format, and started using 3 sections instead of 2. We needed to put our test scores on the same scale. Thus, for all of our samples we used the average SAT score as our metric. The AP score dataframe contained the number of students taking AP tests, the number of students receiving each possible score from one to five, and the population size of the grade 12 year for a certain high school and year. Each of these two dataframes were saved to disk.

The Google Maps API seemed to actually be very accurate and robust to small spelling errors in the high school name, so not much pre-processing was required for this. Sometimes, Google Maps failed to return a distance if it could not compute a driving route between the two locations, for instance when the high school was located on an island. In this case, we simply filled this distance with NaN so that it could be deleted later. This distance data was appended to the rows of the GPA and count data, and then re-saved.

Finally, we had to merge the two datasets together, one containing the GPA, count, and distance data, and the other containing the SAT, AP, and class size data. The issue here was that there was no identifying variable with which we could merge the two datasets, because the variable in the GPA, count, and distance data that we thought was the high school ID number actually was not. Thus, we had to perform a brute force merge based on the high school names. To do this we used the built-in Python “SequenceMatcher” class to find which school names correspond between the two datasets. Before matching, we converted all the names to lowercase, since the matcher is case sensitive, and we removed the words “high” and “school” since they appear  frequently in the school names. We double-checked this matching manually to ensure that the correct rows from each dataset were getting properly matched.

# Modeling and Prediction

We tried to create a model for predicting yields in the University of California system - by year and UC campus, as well as for each high school. We tried to achieve the goal by:

- predicting the yield for each high school
- combining the predicted yields into a single yield for all high schools

In other words, we tried to predict the yield column given other columns (besides the data about enrolled students).

## Choosing the model

We are performing a regression, so that means we should not use classification algorithms such as nearest neighbors, logistic regression, SVMs, or decision trees. Two remaining possibilities would have been to use linear regression or a neural network, but we don't think this problem is complex enough to warrant the added difficulties, complications, and lack of interpretability that a neural network would add. Thus, we decided to use a simple linear regression model. It was a good choice because it was fast to run over our many different experiments that we performed, easy to understand how it behaved so we could better troubleshoot with our rapid development schedule, and descriptive enough to adequately predict the relatively simple problem we were tackling.

First of all, we created a baseline model to which we could compare other models. The baseline model that we used was to simply take the average yield for the past N years for each high school for each campus. For the initial years, if there wasn't enough data to compute this, then the baseline just predicted a yield of 45%, the average yield for all schools for all years.

## Other possible models

- **Using wide datapoints** - one for each year. The datapoints would include the following features:
  - For each school:
    - Number of applied
    - Number of accepted
    - GPA of applied
    - GPA of accepted
    - SAT scores

        - AP scores
        - High school enrollment
        - High school distance from the UC campus
    - Year

  We decided against using that model. One of the main reasons was insufficient time to experiment with combined features (we believe that we should multiply each feature but the number of accepted students).

- **Wide datapoints with random sampling.** - as above, but we would generate random datapoints where only sum of the high schools will be set. This way, we can overcome one of the most significant problems of the model above - an extremely small number of datapoints.
- Using this method above, we should be able to create enough data points for running a Neural Net

## Evaluating our models

To compare how well our models are doing, we prepare a set of evaluation methods. Since our models try to predict the yield ratio for each high school, we evaluate them in two main ways:

- Root mean square as measured for each datapoint in the test dataset
- Accuracy of the prediction extended for the entire academic year

We found extending the predictions to the entire year particularly interesting. We used the predictions for each school to compute the number of enrolled students, and then we divided that number by the total number of admitted students.

## Results

Both our baseline model and linear regression model suggested us that the yield rate would stay at around 45% - 50%, both of which are solid predictions. However, our linear regression model **did** perform better, with an average by-campus error of 68 students, as opposed to the 100 student error of the baseline model. For the ~1000 students who actually did enroll in each campus per year (from these California High Schools, obviously there are more than 1000 students total who start at each UC campus), these two numbers correspond to errors of about 6.8% and 10%, respectively. Thus, our linear model is significantly more useful than the baseline model, and the UC school system could benefit from using it.

The next interesting thing we can look at is our plots of actual vs predicted yield rates for our linear regressor (see Modeling.ipynb). In all of these plots we can see that the points follow a "hump" shape, which means that our model overpredicts for the middle yield rates and underpredicts for both the low and high yield rates. We are not sure what could be causing this error. Perhaps the yield rate is not actually linearly related to any of our features, and we should have transformed some of our features so that we are performing a quadratic fit or similar.

## Issues Encountered

We encountered several problems throughout the process, many of which were unforeseen. For instance, we had lots of trouble merging all of our datasets together. This was because we had high school ID numbers that did not match up, or school names varied by tiny amounts. We also had trouble finding the distances to campuses because of limitations on the number of allowed requests to the Google Maps API. Another issue that we only recently discovered was that one of our original datasets only contained high schools starting with the letters from “A” to “L.” All high schools with names past “L” were not included in one of our datasets for some reason. We decided that this wasn’t too much of an issue because these first schools were probably representative of all California schools, so we merely had a smaller sample size. Moreover, even by incorporating more training features into the linear regression model such as various test scores, we did not see significant amount of improvement it had over our baseline model.

## Future work

Future work would include trying our dataset on different machine learning models besides linear regression to predict yield. In order to work with the distance variable and the AP/SAT test scores data newly added in, we intentionally excluded high schools that are outside California as test scores are only available in California. By doing so, we dropped about 70% of total high schools. In the future, we might need to search for test scores of the high schools in other states just so we will have more data points. So far, our new approach did limit the scope of available datasets. Therefore, the power of our prediction model had also been compromised.

## Summary and Conclusions

In this data analysis task, most of our efforts have been spent into incorporating more new features into the existing datasets like California high school students’ SAT/AP scores as well as the measures of distance between high schools and each UC campus. The reason for trying to merge so much new data was that, from previous iterations, we had found that there’s very limited correlations between features in our old datasets and the yield rate.

Therefore, to evaluate our effort, even though merging new datasets introduced many artifacts; in the meantime, many rows that contain information of Non-Californian high schools were excluded, we might still find potential gain in finding a good linear regression model that would beat our baseline model. And this was our initiative to painstakingly work on more data cleaning and preprocessing.

However, unfortunately, our linear regression model did not outperform the baseline model with a noticeable amount. Given the new dataset we had updated, it would make sense to try on new machine learning models for future studies, and we should extract as much information from our new dataset as possible.