Summary for Metis Project 3

Predicting the Ability of 3rd Grade Students to Meet the State Reading Standard

Sean Davern
Seattle, Fall Cohort, weeks 4-6.5

Overview

This summary is intended to provide Metis staff with supporting information about what and how work was completed for project 3 to aid in the assessment process.

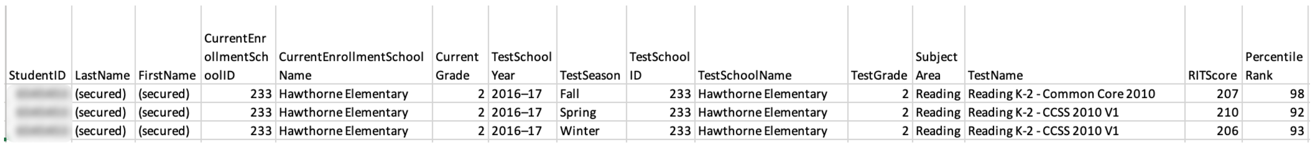
Besides completing the assigned tasks of learning and using classification methods, presenting results and completing the SQL challenges, I also aimed to:

- advance my project documentation and organization practices
- implement the use of python modules in support of Jupyter Notebook analysis
- develop and demonstrate an ability to do more than write SQL to meet challenges

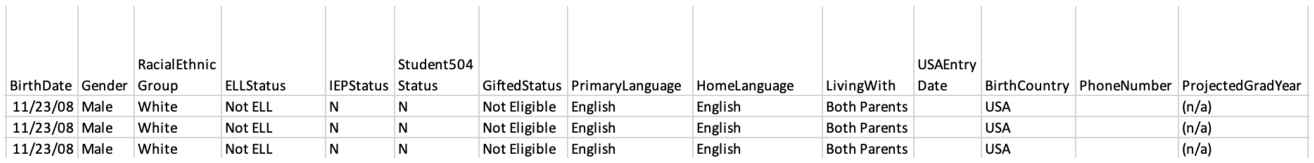
Source data and Feature Engineering

The overall project README.md file shows the number, location and names of the raw data files obtained from Page Ahead. These files came directly from Seattle Public Schools without any processing by Page Ahead. Generally, each where slightly bigger than 20MB in size.

There were 2 types of files. The first provides kindergarten through 2nd grade MAP assessment[1] and demographic data show (sample shown below shows reading data for 1 student from 2nd grade in the 2016-17 school year:



StudentID	LastName	FirstName	CurrentEnrollmentSchoolID	CurrentEnrollmentSchool Name	Current Grade	TestSchool Year	TestSeason	TestSchool ID	TestSchoolName	TestGrade	Subject Area	TestName	RITScore	Percentile Rank
[redacted]	(secured)	(secured)	233	Hawthorne Elementary	2	2016–17	Fall	233	Hawthorne Elementary	2	Reading	Reading K-2 - Common Core 2010	207	98
[redacted]	(secured)	(secured)	233	Hawthorne Elementary	2	2016–17	Spring	233	Hawthorne Elementary	2	Reading	Reading K-2 - CCSS 2010 V1	210	92
[redacted]	(secured)	(secured)	233	Hawthorne Elementary	2	2016–17	Winter	233	Hawthorne Elementary	2	Reading	Reading K-2 - CCSS 2010 V1	206	93



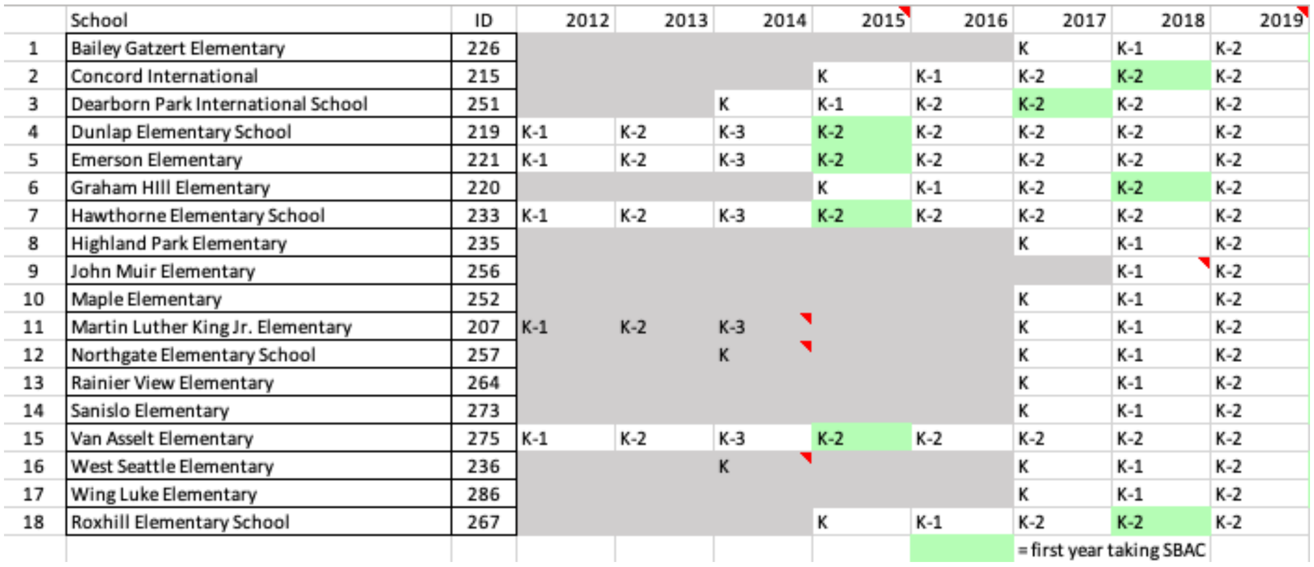
BirthDate	Gender	RacialEthnic Group	ELLStatus	IEPStatus	Student504 Status	GiftedStatus	PrimaryLanguage	HomeLanguage	LivingWith	USAEntry Date	BirthCountry	PhoneNumber	ProjectedGradYear
11/23/08	Male	White	Not ELL	N	N	Not Eligible	English	English	Both Parents		USA		(n/a)
11/23/08	Male	White	Not ELL	N	N	Not Eligible	English	English	Both Parents		USA		(n/a)
11/23/08	Male	White	Not ELL	N	N	Not Eligible	English	English	Both Parents		USA		(n/a)

This data was used to obtain 1st Grade MAP assessment scores, schools attended (TestSchoolName) in each year for determination of if they would have received Page Ahead Book Up treatments and school most attended (a derived feature) and other features (gender, ethnic group, primary language, home language and living situation) . You can see in the above example, the displayed student has 3 test scores in the same school. There is significant variation from school to school for how many test results are obtained (1, 2 or 3) and it is not uncommon for student to move schools, especially poor students. There were 4 MAP assessment files, 1 each for 2015-2018. Combining and initial processing of these files is completed in the ETL Jupyter notebook. The second type of file contains Smarter Balance State assessment (SBAC) scores, is structured similar to the MAP files and there was 1 of them for 2018. The SBAC test is only given once per school year.

The MVP Jupyter notebook shows original development of the python functions to get all records for a student then process them to determine the last available MAP 1st Grade MAP score and the number of Book Up programs they would have received. Those methods are replicated and extended in the Feature Engineering Jupyter notebook for all other features used in the other modeling.

Among the key functions needed to determine if a student would have received the Book Up program was a reduction of another file (Seattle Onboarding History.xslx) provided by Page Ahead showing what schools they served, when they started serving the school and what grade where being served:

Page Ahead Book Up Program Implementation:



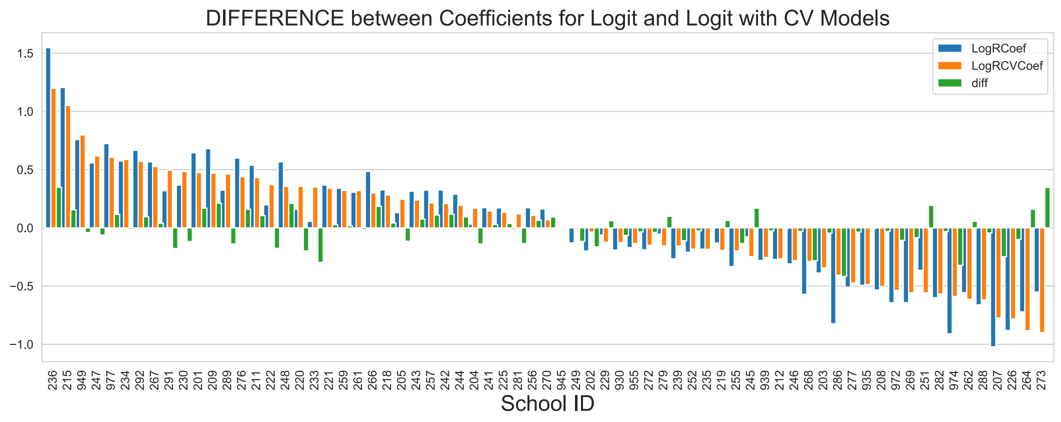
	School	ID	2012	2013	2014	2015	2016	2017	2018	2019
1	Bailey Gatzert Elementary	226						K	K-1	K-2
2	Concord International	215				K	K-1	K-2	K-2 🟩	K-2
3	Dearborn Park International School	251			K	K-1	K-2	K-2 🟩	K-2	K-2
4	Dunlap Elementary School	219	K-1	K-2	K-3	K-2 🟩	K-2	K-2	K-2	K-2
5	Emerson Elementary	221	K-1	K-2	K-3	K-2 🟩	K-2	K-2	K-2	K-2
6	Graham HIll Elementary	220				K	K-1	K-2	K-2 🟩	K-2
7	Hawthorne Elementary School	233	K-1	K-2	K-3	K-2 🟩	K-2	K-2	K-2	K-2
8	Highland Park Elementary	235						K	K-1	K-2
9	John Muir Elementary	256							K-1	K-2
10	Maple Elementary	252						K	K-1	K-2
11	Martin Luther King Jr. Elementary	207	K-1	K-2	K-3			K	K-1	K-2
12	Northgate Elementary School	257			K			K	K-1	K-2
13	Rainier View Elementary	264						K	K-1	K-2
14	Sanislo Elementary	273						K	K-1	K-2
15	Van Asselt Elementary	275	K-1	K-2	K-3	K-2 🟩	K-2	K-2	K-2	K-2
16	West Seattle Elementary	236			K			K	K-1	K-2
17	Wing Luke Elementary	286						K	K-1	K-2
18	Roxhill Elementary School	267				K	K-1	K-2	K-2 🟩	K-2
							🟩	= first year taking SBAC		

This development, resulting in a bookup_dct treatment dictionary, is shown in the BookUpProgram Jupyter notebook. The dictionary is used by the get_treatments python function in use in both the MVP and Feature Engineering notebooks.

Modeling

The MVP notebook contains the original implementation of the sklearn LogisticRegression function. The initial results led to a large amount of EDA, troubleshooting and confidence building to assess the negative coefficient for the nTreatments feature. Subsequent to this, I created the Feature Engineering notebook to expand from the 2 features (1st grade MAP score and nTreatements) included in the MVP. It is designed to pickle the engineered feature and targets data frames which are picked up by the modeling notebook. I next started the Modeling notebook to capture all modeling.

My modeling expansion implemented logistic regression with 5-fold cross validation. I expected to see minimal differences between LogisticRegression and LogisticRegressionCV results. However, I was getting significant differences in coefficients. In particular, I found that a poorly tuned LogisticRegression C hyperparameter could give significantly different school ordering. You'll see graphs in the modeling notebook that enable coefficient comparison which I didn't have time to talk about during my presentation. For example, using C=750 for the LogisticRegression only gives:



While exploring this I found one situations where the schools were nearly ranked backward! I didn't capture that scenario. But, this caused me to be very suspicious of the sensitivity of the results and I spent a lot of time troubleshooting.

In the end I didn't have experience with nearly the number of categorical methods as I wish, but I gained a VERY significant amount of appreciation for understanding a function's default parameters and its hypertuning.

Python Modules

Another goal I had at the beginning of this project was to incorporate the use of cookiecutter and building my project with python modules. I did use cookiecutter and began attempting to implement putting functions into python modules. The notebook modules_attempt.ipynb and modified ./src/obtain.py was the first attempt. You'll see that I didn't figure out how to get my modules and pandas libraries to be in the proper namespace scope for code to successfully execute. I ended up moving on with the hope to connect later with Cliff.

SQL Saga

Another of my original project goals was to use SQL for the whole project based on encouragement from Metis Alums who emphasized its importance. The `Database Creation` Jupyter notebook was the beginning of that. Shortly thereafter, Roberto encouraged me to do the SQL Challenges instead. This was DEFINITELY the right decision. However, because of knowledge of using ORM's from my previous career and ambiguity in instructions for how to use SQL Alchemy, I thought we were supposed to use the full ORM functionality of SQL Alchemy for the challenges. I spent most a whole Saturday doing the first 3 SQL challenge II problems using SQL Alchemy's ORM features.[2] I left those solutions as is. Partly because I didn't know what I was doing, and (I believe) partly because of how the Baseball database was built (without all table primary and foreign keys defined nor created in the correct order) I was getting a LOT of errors and it was taking a LOT of time to troubleshoot. At this point Roberto suggested using SQL Alchemy's core capability only to pass a human-authored query to the database as I did in questions 4-7 and SQL challenge part III.

Finally, please also note that I thought that question 5's direction to use 'CollegePlaying.csv' was an error. SchoolPlayers.csv is what I used for this question. I later learned that other versions of the database had a CollegePlaying.csv. However, the challenge questions directed us to use the database we built on AWS. Those build instructions[3] instruct using `lahman-csv_2014-02-14.zip` which doesn't contain CollegePlayers.csv.