Under the direction and advice of my esteemed TA, the following changes have been implemented to the pipeline:

Processing

1. The read function now accounts for json and excel files along with CSVs. The function can also accept a number of rows to read, should an amount less than the total file be desired.
2. The data exploration graph creation function now adds a title to the plot and takes input columns rather than having one of them be hardcoded.
3. The drop function now takes in a list of variables to drop.
4. The filling function now has options to fill nulls with mean, median and set values.
5. The discretization function now uses the numpy quantile function to derive quantiles. The function also now has options to add a prefix/suffix and an input for what to pre/suffix. Furthermore, the function also accounts for instances when the bins have duplicates – allowing for labels between binary and a maximum of four.
6. A function has been added to account for outliers in a given column.

Classifier

1. The classifier has been massively returned to accept in a list of classifiers, or 'all' to use all available classifiers. This utilized code from Rayid's magic loop.
2. There are now default grids and classifier parameter setting functions.
3. The classifier now records a broad range of evaluation data into a dataframe, which can then be printed to a CSV file
4. Each successful classifier will print a precision recall graph

The subject datasets we drew from include a dataset on several projects proposed by schools and intended to enhance the education of their students as well as a dataset detailing the outcomes of each of those projects. Each project from the former dataset was matched to the latter through a *projectid,* a unique identifying code for each individual project. As such, I could discern which projects had which outcomes. Most notably, the variable we used to determine if a project was a success or not was derived from the outcomes dataset – *fullyfunded*. This binary variable, indicating either true or false, describes whether a project reaches its funding goals and hence was completely funded. Therefore, a true could be considered a successful project, while a false could be considered an unsuccessful project.

To determine what affects the likelihood of a project being fully funded, I chose a number of features.

Feature	Reasoning
'teacher_teach_for_america'	Binary variable describing whether the teacher is part of Teach for America. The additional qualification might aid the project in recognition or trustworthiness.
'teacher_ny_teaching_fellow'	Binary variable describing whether the teacher is a NY teaching fellow. The additional qualification might aid the project in recognition or trustworthiness.

'eligible_double_your_impact_match'	This binary describes whether a project was eligible for 50% off due to a corporate partner. This seems highly likely to correlate to full funding.
'eligible_almost_home_match'	This binary describes whether a project was eligible for 100$ boost due to a corporate partner. This seems likely to correlate to full funding.
'is_exciting'	This binary feature determines whether a feature is considered exciting or not. Naturally, more exciting features would intuitively be more likely to be funded.
'at_least_1_teacher_referred_donor'	This binary feature describes the instance where a donor added to the fund due to the advertisement by a teacher. Naturally, this indicates outreach that would be indicative of a more successfully funded project.
‘at_least_1_green_donation'	This binary feature describes a green donation to the fund. Naturally, a donation is indicative of a more successfully funded project.
'great_chat'	Great chat is a binary that indicates that the project has an active comment chain

	associated with it. It seems likely that lively discussion is correlated with success.
'three_or_more_non_teacher_referred_donors',	This binary feature describes the instance where a donor added to the fund due to the advertisemen. Naturally, this indicates outreach that would be indicative of a more successfully funded project.
'one_non_teacher_referred_donor_giving_100_plus'	This binary feature describes the instance where a donor added to the fund due to the advertisement by a teacher. Naturally, this indicates outreach that would be indicative of a more successfully funded project.
'donation_from_thoughtful_donor'	A binary describing if a curated list of picky ‘power donors’ donated to the project. Given that these donors are likely selective in their donations, this is likely highly correlated with successful projects.
'binnedtotal_price_excluding_optional_support' *	Total price excluding tips. Likely the higher this amount is, the harder it would be to fund.
'binnedtotal_price_including_optional_support' *	Total price including tips. Likely the higher this amount is, the harder it would be to fund.

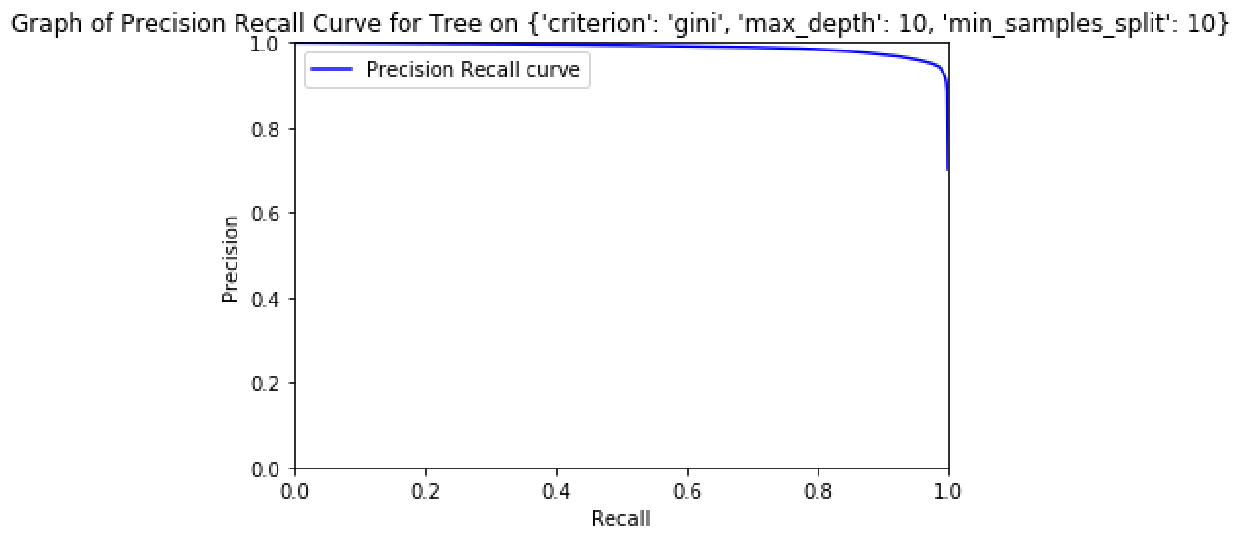
'binnedstudents_reached' *	Number of students impacted by the project. The greater the amount, it is possible that it was more likely to get funded.
'binnedteacher_referred_count' *	This continuous feature describes the number of referrals. Naturally, the higher the quantile of referrals, the more likely the project is to have been funded.
'binnednon_teacher_referred_count' *	This continuous feature describes the number of referrals. Naturally, the higher the quantile of referrals, the more likely the project is to have been funded.

*Binned variables were simple derived from the original feature, using quantiles to assign numerical labels. For instance, for feature ‘price_excluding_optional_support', a price in the highest quantile would be listed as ‘4’ while one in the lowest would be listed as ‘1’.

Originally, my subject variables included information on the school itself – such as whether it was a year round public school or magnet school or such. However, I found that there was no real increase in prediction accuracy from utilizing this descriptive features and so they were dropped.

In terms of our analysis, I found that a tree classifier with parameters: {'criterion': 'gini', 'max_depth': 10, 'min_samples_split': 10} derived the highest AUC ROC of around 0.98. Overall, tree classifiers seemed to perform the best across all metrics on average, in precision, accuracy, recall and F1. Even in terms of duration of running time, tree models executed the most quickly. In comparison, KNN classifiers ran quite slowly, with one KNN with parameters: {'algorithm': 'auto', 'n_neighbors': 50, 'weights': 'uniform'} taking around 184 seconds to run.

Below is the precision recall graph of the best performing classifier by accuracy. It is clear that both precision and recall were also high for this classifier with the listed parameters.



model_typ	parameters	duration	accuracy	precision	recall	F1	Average Precision	AUC ROC Score
Tree	{'criterion': 'gini', 'max_depth': 10, 'min_samples_split': 10}	0.559449673	0.947467	0.944101	0.983426	0.963362	0.988211351	0.97837495
KNN	{'algorithm': 'auto', 'n_neighbors': 50, 'weights': 'uniform'}	183.5991843	0.941149	0.936615	0.982707	0.959108	0.984156048	0.974615382
Forest	{'max_depth': 5, 'max_features': 'sqrt', 'min_samples_split': 5, 'n_estimators': 100}	7.947753429	0.934341	0.925569	0.985782	0.954727	0.985586831	0.972225706
Boosted	{'algorithm': 'SAMME', 'n_estimators': 1000}	85.71586728	0.93123	0.79828	0.998687	0.887308	0.979825296	0.960253293
Logit	{'C': 1, 'penalty': 'l1'}	1.885953665	0.9241	0.939729	0.953051	0.946343	0.979169914	0.960083735

Above are the best performing parameters by model. As can be clearly seen, tree is the most well-rounded of the classifiers, though Logit and Forest also performed quite well in this regard. As such, I would recommend to any individuals working on this project about projects to rely on Tree, Forest or Logit classifiers in their analysis. Accuracy overall was quite high across all the classifiers, which suggests that what we learned in class (that accuracy is a poor metric) is true.