**RESEARCH ROADMAP**

**Data**

1. grades\_cleaned-20170520.csv
   1. what: cleaned grades data
   2. where: Regression Analysis data directory
   3. other info: does not contain anonymized student ID
2. grades2430.xls
   1. what: unclean grades data
   2. where: EDM/Projects/EWS/data/Yilei\_self\_assessment/
   3. other info: contains anonymized student ID
3. self\_assessment\*.xlsx
   1. what: self assessment data, 1 spreadsheet for each exam, 3 total
   2. where: EDM/Projects/EWS/data/Yilei\_self\_assessment/
   3. other info: contains anonymized student ID
4. survey.xlsx
   1. what: survey data from Yilei's end of semester questionnaire
   2. where: EDM/Projects/EWS/data/Yilei\_self\_assessment/
   3. other info: contains anonymized student ID

**Questions that need answering**

1. How many students satisfy all of the following conditions?

- Completed course

- Completed all self-assessments

- Completed Yilei's survey

If this number is above 40, we can continue with the study.

**Producing one giant table**

We need a way to consolidate all the data from the different tables.

1. A particular challenge is the fact that grades data has already been cleaned but does not contain anonymized student ID's. To save ourselves the trouble of having to re-clean grades data that contains student ID's, a better solution would be to merge student ID from the unclean data into the clean data. Assuming that all total\_weighted grades are unique, you can drop all columns from the unclean grades table except for student ID and total\_weighted grade, and perform an [INNER JOIN](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.join.html) between this smaller table and the cleaned grades table, joining on the total\_weighted column.
   1. Note: total\_weighted in the cleaned grades is 100 multiplied by total\_weighted in the un-cleaned grades, so one column must be transformed to be consistent with the other.
   2. Note: If total\_weighted grades are not unique, then perform the INNER JOIN on two columns, total\_weighted + some other column with low likelihood of having repeated values.
2. Assuming student ID’s can be incorporated successfully into the cleaned grades table, then the remaining tables can all be joined against this grades table using student ID.

**Correlation Matrix**

Let *A* be the giant table produced from above. It will have many features, some numeric, some [categorical](https://en.wikipedia.org/wiki/Categorical_variable) (features where the possible values have no inherent ordering), and some [ordinal](https://www.ma.utexas.edu/users/mks/statmistakes/ordinal.html) (features with non-numeric values but that have some ordered relationship). Make a filtered table *A’* that excludes all categorical features and re-encodes ordinal features as numeric features. Now you can make a [correlation matrix](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.corr.html) and plot it using [Seaborn’s heatmap](http://seaborn.pydata.org/generated/seaborn.heatmap.html).

* Note: Most likely, this matrix is going to be incredibly large and unreadable. You’re going to want to build smaller sub-matrices for closer examination.
* Note: Play around with combining multiple columns into one via an aggregation (average, standard deviation, min, max, etc...)

**Learning feature importances**

Let *A* be the giant table/matrix. We are interested in using algorithms to learn the importance of various features for predicting the final total\_weighted grade. Thus, we need to split *A* into a feature matrix *X* (everything in *A* except *y*) and a label column *y* (total\_weighted grade).

* Note: *X* contains categorical features that will need to be one-hot encoded.
* Note: *X* also contains ordinal features that will need to be re-encoded.
* Note: *X* will have missing values for blackboard quiz grades. You can replace these null values with either 0’s, the quiz mean, or the quiz median.
* Note: Predicting the total\_weighted grade from a grade late in the course is not very interesting. Consider only leaving early grades in *X* (like the first two or three homeworks and the first midterm).

If we exclude late-stage grades from *X*, my expectation is that no algorithm will be particularly accurate. While we should measure the performance of these algorithms, we’re more interested in what features the algorithm is focusing on in trying to make a prediction. Here are some algorithm choices:

* Linear regression: The sklearn LinearRegression model has an attribute [**coef­\_**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html). After you’ve fit the model, coef\_ shows the coefficients the model has selected for each feature. The larger the absolute value of the coefficient for a feature, the more important that feature is.
  + [Lasso regression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html): This is a variant of LinearRegression that tends to shrink coefficients of unimportant features to 0. It is frequently used as a feature selection tool.
* Decision trees and random forests: These models have an attribute called **feature\_importances\_** that provide similar information to linear regression coefficients.
* [Linear SVM](http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html): Like LinearRegression, this model provides coefficients.

An important consideration to remember is that if two features are highly correlated with one another and contain redundant information, a model may choose to weigh one feature heavily and not consider the other feature at all due to its redundancy. For this reason, it may also be useful to try different subsets of features for training and feature importance evaluation.