# COURSE MEDIAN PREDICTION VIA SYLLABI ANALYSIS [MILESTONE REPORT]

CORALIE PHANORD, GRAESON McMAHON, KELSEY JUSTIS

February 17, 2015

## 1 RECAP

### 1.1 Problem Statement

We are applying supervised machine-learning algorithms to course syllabi for the extraction and discovery of content features that play a role in predicting the corresponding median grades.

### 1.2 Milestone Goals

Below are the goals as defined in our initial project proposal.

#### 1.2.1 GOAL 1: Collect and format data for processing by January 29

Measure of success: Data source and format choice are finalized, data is (largely) ready to process.

#### 1.2.2 GOAL 2: Flexible parser to extract features from syllabi developed by February 12

Measure of success: Code is developed and capable of scanning syllabi for features in raw text and document formatting.

#### 1.2.3 GOAL 3: Initial development of chosen algorithm started by February 17

Measure of success: Algorithm used to learn features from syllabi via parser is chosen. Early work is started with emphasis on code structure outside implementations of chosen algorithm are examined for best practices.

## 2 MILESTONE PROGRESS

We are happy to report the successful completion of all our milestone goals. Please read below on how these accomplishments were made possible.

### 2.1 Data Collection

Data collection was a lengthy process and involved two primary sources: Dartmouth class/department websites and department heads themselves. By searching the former and reaching out to the latter, we managed to accumulate upwards of 500 syllabi, largely in .pdf format. 30-40% of these were unusable, either because their respective courses medians were not listed online or they did not contain parsable text (several were scans of physical syllabi).

Median grades were taken from the registrars website[1] and placed into a .csv file using Excel.

## 2.2 Data Formatting

After collection, we used Xpdf's[2] pdftotext program to convert every .pdf file to .txt to facilitate later parsing. Each of these .txt files was renamed using the following scheme: DEPARTMENT-COURSE #-TERM-SUBCOURSE #, closely following the registrar's course-naming system. We then wrote code in MATLAB that, given the title of a syllabus .txt file and the columns of the median .csv file (term, class name, and median), output a syllabus's corresponding median. This was error-prone, as small inconsistencies in the registrar's course-naming conventions made our code generate a large number of false negatives when attempting to match syllabi titles to entries in the .csv file. Manual examination of each unmatched syllabus was necessary, preventing us from completely finishing data preprocessing. As a result, we used about 235 syllabi in our milestone tests. We anticipate later tests will include around 100 more.

## 2.3 Syllabi Text Parser

Upon successfully collecting and formatting the syllabi we were then able to parse through the .txt files and begin examining the more than 450,000 words of content. The final parser design was decided upon reviewing best practices found in others' code online. One script that proved especially impactful was developed by Suri Like.[3] Using heavily modified portions of this work we filled in the skeleton of our program that filters through the text content for alphanumeric words. This parser is then capable of producing the dictionary of vocab used in each individual syllabus. The parser was then further extended to handle a desired batch of syllabi .txt files and enable feature extraction.

## 2.4 Syllabi Feature Extraction

Although our decision process regarding which features are most relevant for the prediction algorithm's success is ongoing, we have started examining a simple set of high-level and low-level features that include:

*High-Level:* Course syllabus department, number, syllabus length, enrollment, and term offered.
*Low-Level:* Number of stop words[4] used, number of percent signs present (as an indication of the grade-breakdown), presence of labs, negative word count (using a short list of words we were initially interested in)

For our milestone tests, only course number, enrollment, syllabus length, and negative word count were finished and included in the algorithm tests. In the near future we will test more of these features and incorporate recently found online databases for discovering additional low-level features. These databases might include Princetons WordNet[5] for word and phrase connotations, and various combined databases of stop words to shift the algorithms focus to more meaningful words/phrases.

## 2.5 Algorithm

After much deliberation, we decided upon the use of a decision tree-based regression learning algorithm for training our data. We reasoned that since our data consists of both categorical and continuous numerical data we would seek a model that optimally addressed both. Our search for such a model led to Ross Quinlan's C4.5 algorithm[6], which can create a decision tree incorporating both data types.

[1]http://www.dartmouth.edu/ reg/transcript/medians/
[2]http://www.foolabs.com/xpdf/home.html
[3]Suri Like's Wordcount.m Script
[4]ranks.nl stopword listxpo6.comGoogle Stopword Project
[5]http://wordnet.princeton.edu/
[6]http://en.wikipedia.org/wiki/C4.5_algorithm

The C4.5 algorithm is often used in classification problems, for which it is ideal to create decision nodes based on the information gain or entropy decrease from choosing to split the data at a certain feature. Since we are focusing on prediction rather than classification we have designed our algorithm specifically for regression. Thus, we have used standard deviation reduction to find the feature and threshold providing the best split.[7]

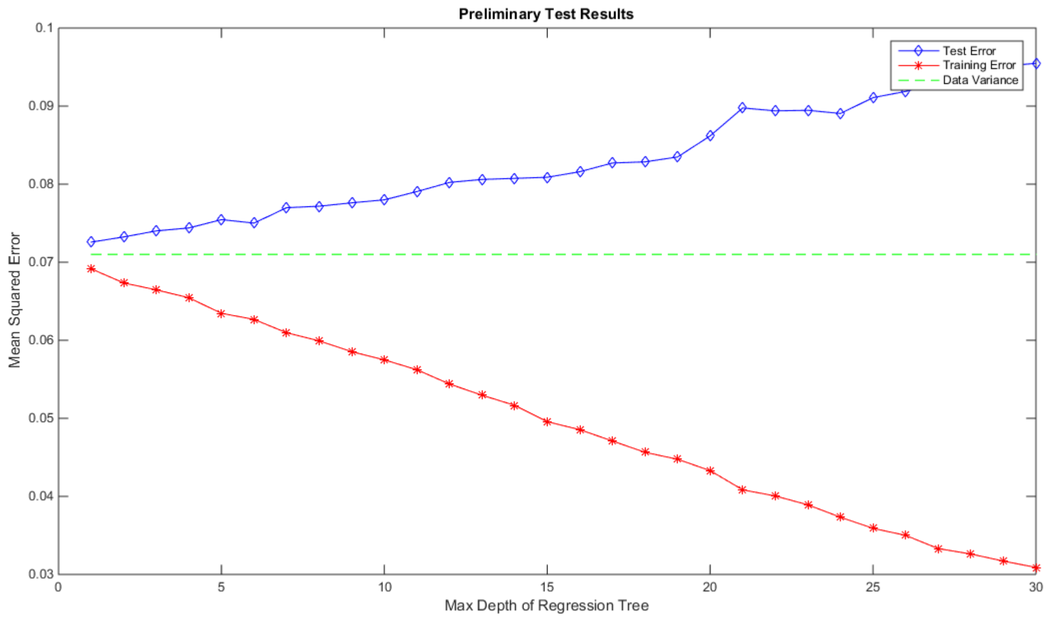
Our stopping criteria include: when the standard deviation at a node falls below a preset threshold, when the current depth equals the maximum depth, or when the sample size at a node is beneath some constant. Below is the pseudocode for our implementation (given $m$ training examples with $n$ features):

1. Check if stopping criteria are reached; if so, create a leaf node which predicts the mean value of the examples remaining at that node. Return this node.
2. For each (numerical) feature $A$, sort the data according to $A$ and try all meaningful partitions $(1 : m, m+1 : end)$, $m \in [2, m-1]$. Find the partition for which the sum of the standard deviation of each subset is minimized.
3. Select the feature $bestA$ and partition which produces the smallest total standard deviation, and create a decision node $N$ encapsulating $bestA$ and its ideal partition.
4. For each sublist created by splitting on $bestA$, recurse on that sublist and add the result as a child for $N$.

Currently the algorithm works with numerical data only, but the functionality for mixed numerical and categorical data will be added for the next test.

## 2.6 Results

To test the algorithm, we randomly partitioned the dataset into training and test sets (70% and 30%, respectively) and built trees with maximum depths from 1 to 30. As previously stated, we worked with 235 examples and 4 basic features (course number, enrollment, syllabus length, and negative word count). The following graph shows the average mean squared error at each maximum depth (averaged over 10 random partitions).



[7]http://chem-eng.utoronto.ca/ datamining/dmc/decision_tree_reg.htm

As expected, the training error declines as we increase the maximum depth of our regression tree. Unfortunately, the increasing test error suggests that we are overfitting at even the lowest depths. The variance of our target data is displayed in green, implying that our algorithm currently offers no advantage over simply predicting the mean for every test example. We hope to improve upon these results in a number of ways, including updating our feature set, implementing additional components of C4.5 such as post-tree-construction pruning, and more robust prediction at leaf nodes.