ABSTRACT

A Baseline Admissions Prediction Model
With Textual Analysis and Confidence Interval Estimations

Stephen Ryan Beckham, M.S. Eco.

Mentor: Van Pham, Ph.D.

Essays submitted out individuals applying to Baylor University may contain hidden information that would assist the admissions department in their decision to accept the appicant. Through textual analysis, this paper attempted to reveal signals of a student's intent to attend Baylor if accepted as well as offering an additional platform to judge a student's ability. Both results are independent of other information gathered from the application process. The models created were found not to be strong enough to act as a stand-alone decision rule. However, the new variables created can be used in Baylor's admission model to increase its effectiveness. The groups of words in commitment, Baylor and admissions groups all prove to be influential, which can be used in other parts of the enrollment process, such as phone interviews. This paper also simulates a confidence range around yield estimates generated from the current model being used at Baylor.

A Baseline Admissions Prediction Model With Textual Analysis and Confidence Interval Estimations

by

Stephen Ryan Beckham, B.B.A.

A Thesis

Approved by the Department of Economics

_____

Thomas M. Kelly, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of

Master of Science

Approved by the Thesis Committee

_____

Van Pham, Ph.D., Chairperson

_____

Scott Cunningham, Ph.D.

_____

Chris P. Pullig, Ph.D.

_____

Dean M. Young, Ph.D.

Accepted by the Graduate School

August 2014

_____

J. Larry Lyon, Ph.D., Dean

Page bearing signatures is kept on file in the Graduate School.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER ONE

Introduction

College admissions departments have a dual mandate given to them. First and foremost they must admit enough students to meet the budgetary needs of the university. During this process, the department must also maintain a certain level of academic excellence. Students admitted to college are expected to perform at a level to graduate and become high achievers in the marketplace. Baylor is a private university that is not endowment based, therefore the majority of income depends on tuition. Missing the needed enrollment by even a small percent could end up costing the school millions of dollars. If too many students are enrolled, classes become overcrowded and school resources cannot be properly allocated.

The process is made difficult by the desire of the university to increase their rank in publications such as *U.S. News & World Report.* Part of the national ranking system for a university is the yield rate of admittance. This rate is the percentage of students that enroll in the school compared to the number of students that are accepted. A student attending a university is a two stage decision. If a student has applied to a university, that school has the choice to either accept or deny the application. Once an acceptance is offered, the student then has the choice to enroll or not. Many students fill out applications for several universities in order to find the most prestigious school that will accept them along with the best financial aid package. Because of the indecisiveness of the students applying, admissions departments need a better model to sort out applicants who truly want to be Baylor students. The expected yield estimate is vital because it determines how many students will need to be accepted in order to meet the budgetary needs. This has become increasingly difficult over the last fifteen years as applications for each school

have increased significantly and yield rates have been falling as students apply to multiple colleges. (Jaschik, 2012)

The yield rate exemplifies the importance for a university to admit the right candidates. If the admissions department had a better way to predict enrollment, they could lower the amount of acceptances they issue and in turn raise their yield rate. It is hard for the admissions department to know both a student's ability and their probability of enrollment if admitted due to adverse selection. Potential students want to be admitted in order to receive the benefit of having a prestigious college diploma despite their known inabilities. This problem is the very foundation of adverse selection, as the students will only give the information that will highlight their abilities and not the information that could potentially prevent them from being admitted. This information is not limited to academic deficiencies, but desire to attend as well. A university could be a student's fourth option, but they will not share that information in order to get accepted and have as many options available when it is time to make their decision.

Admissions departments need several different metrics to rate a student on both probability to enroll as well as ability to perform at a high level once enrolled. The standard way of rating a student's ability has been the SAT, ACT, high school rank and other measurable metrics that are based on performance in high school. Another way of finding this hidden information can be through the essays a student writes. For Baylor University, an applicant is asked to complete two essays: one regarding why the student is choosing Baylor and the other regarding what the student is looking for in a college. These simple questions can be full of signals on how committed the student is to coming to Baylor as well as being another measurement of their ability.

The methodology that is used also has other benefits for the admissions department. The most important of these are the results from the logistic regressions done to predict enrollment. The regression coefficients show the importance of certain groups of words used to predict enrollment. The higher the coefficient, the more effect that group of words has on the outcome. The group of words provided by the admissions team, as well as the Baylor specific words and commitment words are all very influential in the probability. These words can be changed around to find better combinations to improve the results. They can also be used in other forms of recruiting, such as the phone interviews. Baylor conducts phone interviews with certain recruits to persuade a prospective student to enroll and also to gage their interest. Since the group of words used in the regressions proves to be influential, the phone interviewers now have new words that can be clued in on during the conversation to provide additional information on the prospective student.

CHAPTER TWO

Literature Review

The ability to write well is a difficult task involving skills developed in grade school. It requires memory, language, and thinking abilities. A person is stressing their memory when performing extended writing because it requires you to remember what you have done and where you are going. (Kellogg & Raulerson, 2007) These types of cognitive skills are needed in higher education and application essays offer a platform for a college to measure these abilities. Studies at the University of California have even shown that a student's writing ability is the best predictor of first semester achievement. (Geiser & Studley, 2001)

Essays were a major component in college admissions until the 1970's. During that decade, college admissions began to rapidly increase and it became too expensive to properly analyze the essays. Multiple choice tests were found to use testing time efficiently and were able to cover a large group of concepts. (Bridgeman, 1991, pg. 320) At the same time as the Cold War, natural language processing was a governmental research task. It was felt that being able to process text and comprehend it at a binary level could offer an advantage over Russia. With the end of the Cold War, the government ended the program and it was privatized. Now this field is run by companies like ACCUPLACER who offer "reliable, valid and accurate" reports to universities and high schools for automated essay scoring (AES). (Ericsson & Haswell, 2006)

Recently, there has been a movement to remove the SAT as the primary benchmark for college admissions. SAT preparation and review has taken over classroom settings and taken time away from core classes. Universities define what is important, which influences what is taught in high school. Supporters of this movement want to implement comprehensive curriculum achievement tests to replace SAT testing. This would take away the emphasis of the SAT by high school teachers and place more emphasis on such areas as history, geometry or language arts. (Atkinson, 2011, pg. 5)

Computer grading of essays and the use of AES has grown over the last ten years. AES has been a widely debated part of the higher education selection process. In order to use AES, a group of test essays are hand graded and commonalities are found in both the structure and language used in the essays. The computer is then programmed to find correlations in the papers being tested to assign a grade. (Dikli, 2006) Careful consideration needs to be taken for each usage of AES. Papers written on different topics will have different language that is acceptable. Because of this, latent semantic analysis (LSA) is used to specialize the scoring. A word or phrase can take on a specific meaning in one passage and have a different meaning in another. There is also the potential for two passages to have the same words and phrases, but have completely different meanings. (Landauer, Laham, & Foltz, 1999) Each school should have its own modeling to develop its own specificity.

This research creates two groups of variables from the application essays. There are grammatical variables that measure a student's ability to write and signal variables that attempt to uncover the student's intent to enroll in Baylor if they are accepted. This approach can help the admissions department by both finding more qualified applicants as well as trying to raise the yield rate. This also helps overcome the biggest criticism of AES, which is that it does not take into account meaning.

Mechanical checks cannot properly assess discipline-based assignments that need to show a substantial knowledge in the field. (Ericsson & Haswell, 2006)

The time and resources it would take to read every entrance essay is beyond any reasonable cost. Computer programs are the only efficient way of using data collected from essays effectively. This process also removes the human element of having a grader read and try to comprehend what the essay intends to convey. AES offers consistent grading, a major problem with human evaluation. The essay simply becomes a collection of variables lacking any human emotion or preconceived ideas. However, the size of the data has to be taken into consideration and a balance needs to be found between the two. We have to consider the applications as just objects and the words within them as input strings. By using a computer, the challenge is providing the necessary background knowledge and allowing for a large enough database for association. (Page, 1968, pg. 211)

CHAPTER THREE

Data Overview

The raw data was provided by the Admissions Department of Baylor University from previous applicants. Table 1 shows the number of observations available for the time period 2011 - 2013

Table 1: Number of Observations

Year	# Students Accepted	Total # Students Submitting Essays	Total # Accepted Students Submitting Essays
2011	15430	4612	1068
2012	16879	6257	1495
2013	16809	9301	1985

The main issue that arises from the textual analysis is the low participation rate of students submitting the two essays, because Baylor has chosen to make them optional. In 2011, only 7% of the accepted students submitted essays. There was a surge in 2013, but the rate remained at only 12%. The choice to participate in the essay section could be seen as a signal itself as a true interest in the university. Unfortunately, the low percentages make it unrealistic to use. If the school could guide students to participate in the essay submission, there might be gains seen in the predictive cababilities.

This paper focuses on the question of whether the textual analysis can be a stand-alone decision model. Because of that purpose, none of the other variables included in the raw data such as high school rank, distance from Baylor, etc. were being used in the regressions. All variables used were created through Python. The variables created fall into two categories. The first category of variables measures the student's ability

and focuses on grammatical counters. The second category of variables includes words that can be considered signals for interest in Baylor.

The variables in Table 2 were used to determine the quality of writing in the essays. The quality of writing was used as a way of predicting an applicant's true ability that may not have been captured in standardized testing. Giving students a different test at the university would be another option, but this would drive away most applicants. It is unknown if there are any other cost efficient way of measuring a student's ability. Writing utilizes higher level thinking skills which can showcase knowledge. In order to have a baseline to predict ability, first semester GPA was used.

The other area of measurement was the applicant's commitment to Baylor. Several different variables were created to measure a student's desire to enroll in Baylor if admitted. Two areas were focused on to find a student's probability of enrollment. These two areas were their expressions of faith and their knowledge of the campus and traditions. Being a faith-based campus, a student with strong religious beliefs may be drawn more to Baylor than competing state universities. Knowledge of Baylor shows that the student has either grown up learning about the university, which signals a legacy, or they have taken time to learn the information, which shows interest. Table 3 shows all variables considered a potential signal for a student's interest in Baylor, as well as a description of each.

Table 2: Ability Variables

Variable Name	
Essay1Len	The total number of words used in the first essay
Essay2Len	The total number of words used in the second essay
TotalLen	The combined number of words used in both essays
GrammarWrong	The percentage of words in both essays which were misspelled
WordLength	The average length of all words in both essays
NumUnique	The number of different words in both essays
Lexrich	The percentage of unique words to the total number of words in both essays
Syllables	The total number of syllables used in both essays
Sentences	The total number of sentences in both essays
fleschreading[1]	A readability test. The lower the score, the higher the expected comprehension level for the essay
fleschgrade[2]	A readability test that shows the expected grade level that will comprehend the essay

[1] Flesch Reading Score = 206.835 - 1.015(total words / total sentences) - 84.6(total syllables / total words): 90.0-100.0 easily understood by an average 11-year-old student, 60.0-70.0 easily understood by 13- to 15-year-old students, 0.0-30.0 best understood by university graduates

[2] Flesch - Kincaid Grade Level = 0.39(total words/ total sentences) + 11.8(total syllables / total words) - 15.59

Table 3: Commitment Variables

Variable Name	Description
commitmentcounter[3]	The number of words in both essays used by people who did come to Baylor
commitmentpercent	The percentage of words that are commitment words
faithcounter[4]	The number of words in both essays used to express their faith
faithpercent	The percentage of words that are faith words
baylorcounter[5]	The number of words in both essays used to signal their knowledge of Baylor
baylorpercent	The percentage of words that are Baylor specific
totalbible	The number of times in both essays that a book of the Bible was referenced
biblepercent	The percentage of words that are books of the Bible
admincounter[6]	The number of times in both essays in which words given by the admissions team were used
adminpercent	The percentage of words that are from the admissions team

[3]Words used: pride, visit, leader, unity, campus, legacy, important, environment, only, home, comfort, mom, dad, mother, father, family

[4] Words used: shall, unto, lord, thou, thy, ye, God, son, hath, Israel, king, people, house, before, children, against, shalt, land, day, hand, behold, faith, sons, hast, o, over, she, David, great, Jesus, thine, father, neither, give, take, am, forth, brought, name, away, pass, two, according, days, city, earth, Moses, thereof, whom, know

[5] Words Used: Baylor, bears, bear, Waco, oso, quadrangle, ivy, burleson, hankamer, Louise, Herrington, truett, dia del

[6]Words Used: top, choice, first, dream, dreamed, always, growing, grown, grow, wonderful, visit, blessed, experience, passion, passionate, tradition, atmosphere, values, perfect, heart

CHAPTER FOUR

Methodology

General Textual Analysis

Different forms of regression were used to predict GPA and enrollment based upon acceptance. Linear regression was used to predict GPA using variables created through textual analysis. Logistic regression was used to predict a student's probability of enrolling based upon acceptance. The logistic regression returns the log odds, which will be a number between 0 and 1. The log odds can be converted to a probability, which shows the chances of enrollment if a student is accepted.

In order to collect and analyze the data, the research method included the use of two programming languages; STATA and Python. Python is a scripting language that is preferred for the use of unstructured data analysis. The essays are imported into Python and then parsed. The output is stored into a file which is imported into STATA, a statistical software package, for analysis.

In order to create the model, stepwise regression was implemented. Stepwise regression is typically considered a poor approach by econometricians due to its limitations. These limitations include biased p-values and R^2 values as well as potentially leaving collinearity problems. There is also a lack of analysis on what goes into the model, making it harder to justify the results. In this research, many of these problems were deemed capable of being overlooked. One assumption that is made is that the covariates in the logistic regression are independent. The R^2 is not statistically sound with logistic regressions so a Pseudo- R^2 is used. STATA uses an adjusted Cox & Snell method for goodness of fit that is used as a replacement for the standard R^2 is used for in linear regressions.

The goal of the model was accurate prediction with causality used as a robustness check. The reason why students come to Baylor was not the goal. The objective remains predicting whether they will come to Baylor if accepted, and what type of characteristics they might have. With the long list of variables that was made available, future work can be done to find causal reasons, but the results of this testing show that finding causality could be difficult.

The STATA program used to perform the stepwise regression was vselect. This program allows the user to perform the regressions with either forward selection or backwards elimination. Akaike's information criterion, Akaike's corrected information criterion, Bayesian information criterion and R^2 adjusted are all criterion outputted using the vselect command. This allows the user to select from a group of models that could potentially work for them. The models for enrollment prediction and GPA were chosen from the four potential models created. In order to choose the best of the options, 50% of the data was used as a training set and the model that had the best prediction power was used. Different criteria were used for the different predictions.

Predicting Probability of Enrollment

The method used to predict a student's probability to enroll if accepted was a logistic regression. The decision on whether a student comes to Baylor is a binary decision. You either enroll, or you do not. A logistic regression can predict the probability to enroll based upon acceptance. If the admissions department can determine which students have a high probability to enroll based on their self-identification through the essay, they can increase the yield rate. This is done by selecting students who are identical in testing metrics, but have signaled their desire

to enroll. The textual analysis was done to try to identify what students could say that would aid in the decision of whether or not to accept them is attempted.

The variables in Table 2 and Table 3 were both used in the prediction of probability. The variables used to judge the writing are important because there is a middle ground of ability that Baylor has to focus on. There is a standard which writing needs to be above in order to prove they have the ability to compete at the college level. There also seems to exist a point where the student's writing quality is so high, they are probably getting offers from more prestigious schools. The variables in Table 3 were all created with the idea that they can signal a studentôs desire to attend Baylor University. Each of the variables considered a different aspect of Baylor life and attempted to quantify a student's interest in each of them. The commitment words came from high frequency words that were unique to students who enrolled in Baylor compared to words used by students who were accepted and didn't enroll. The faith words were high frequency words in the King James Bible. This version was chosen because it uses the language used in most quoted verses. Not often is modern language used to quote scripture. There also needed to be a count of Baylor specific word counteract students who copy and paste essays. The Baylor word counter rewards students who focus on the university they are currently applying for. These were unique words taken from websites of Baylor, Texas A&M, University of Texas (UT), Texas Christian University (TCU), Texas Tech and Southern Methodist University (SMU). These schools were chosen because they are the main competitors for students interested in Baylor. The traditions pages of these schools were scraped and vectors of words collected then compared to find words only on Baylor's traditions page. The traditions pages were chosen because they collect the most university specific words and messages of the websites. Mentioning

these words shows knowledge of Baylor specific terms and can signal interest. The administration words were chosen by the admissions team from Baylor University. They have experience in high frequency words from past students that enrolled that may not have shown up in the other variable categories.

The textual analysis portion attempted to create a stand-alone decision model of only the essays. Ultimately, information gathered from the essays can be used in a larger model as a way of increasing its predictive power. This methodology only used the textual analysis to demonstrate its stand-alone predictive power. For robustness, three models returned from the stepwise regression were used. Each model included the variables as well as some interaction variables that were created to help find the true form of the relationship. Table 4 shows the variables used in each of the different models.

Each of the models was trained on a random 50% of the data and tested on the other 50%. Because of the small amount of enrollments with essays submitted in the data set, the 50% mark was chosen to allow a legitimate amount of training and testing. The testing predicted the probability that each student would enroll in Baylor if accepted. The models have a bias toward lower probabilities and because of this, the selection point was set to 30%. This selection point has been found in other studies to be a viable point. (Ruznik & Dawes, 2001) This selection point represents the probability at which a student should be accepted or rejected. Any student that received a probability of at least 30% was classified as someone who would enroll. For the results, each of the observation's actual result was compared to the predicted result to get a percentage for the correct predictions.

The effectiveness of the model will be the rate at which students enroll compared to their predicted actions. The robustness check was the actual results, which is comparing who attended Baylor versus who was predicted to. As a stand-alone model, this could help in making marginal decisions. However, an effective use of the newly created variables would be to include them as potential additions in a larger model that includes high school metrics.

Table 4: Variables used in Models to Predict Enrollment

Method Used	Variables Used	Interactions Used
R^2	commitmentcounter	faithcounter * baylorpercent
	baylorcounter	$baylorcounter^2$
	totalbible	$commitmentcounter^2$
	syllables	$totalbible^2$
	admincounter	TotalLen* WordLength
	Essay2Len	biblepercent + baylorpercent + faithpercent +commitmentpercent
	baylorpercent	GrammarWrong + commitmentpercent
	NumUnique	
AIC	commitmentcounter	faithcounter* baylorpercent
	baylorcounter	$baylorcounter^2$
	syllables	TotalLen* WordLength
	admincounter	
	Essay2Len	
	faithpercent	
	GrammarWrong	
	NumUnique	
BIC	commitment counter	faithcounter * baylorpercent
	baylorcounter	
	syllables	
	admincounter	
	NumUnique	

The method used to predict a student's GPA was linear regression. The assumption being made was that there is a linear relationship between the effectiveness of a studentôs writing and their potential first semester GPA. The ordinary least squares (OLS) linear regression model created using stepwise regression was

$$gpa = \alpha + \beta_1 syllables + \beta_2(NumUnique * lexrich) + \beta_3(fleschreading * GrammarWrong) + \beta_4 sentences^2 + \beta_5 TotalLen^2 + \beta_6 GrammarWrong + \beta_7(syllables * TotalLen) + \beta_8 sentences + \beta_9 fleschreading^3 + \beta_{10}(fleschreading * fleschgrade) + \beta_{11}(fleschreading * fleschgrade * GrammarWrong) + \beta_{12}(sentences * WordLength) + e$$

The model that had the best prediction power uses AIC. The amount of variables used in this format would be less than models using the R^2 method, since AIC balances between goodness of fit and complexity of the model. Due to the smaller number of variables used, collinearity is also reduced. This also helps avoid the problem of overfitting. With too many variables, the model would be too specific to the single application and would not have relevant predictive power for other models that it is applied to.

The group of variables used is listed in Table 2. Each of the variables was selected because of their potential help in grading the writer's ability. Longer, more complex words are captured in the syllables and word length variables and show a mature vocabulary. The lexical richness and number of unique words show a large vocabulary and the ability to use it. The sentence length and number of sentences represent the ability to convey more in depth ideas and follow up on those ideas.

The data set was broken into six separate groups, based upon SAT scores. If the ACT was taken instead of the SAT, a conversion score was calculated to produce the equivalent SAT score. These scores have historically been the standard used to

predict a student's ability. In order to show that this model can complement the SAT, sub groups of SAT scores were created to hold that variable constant. The assumption is being made that students within the sub group SAT are similar. This helped control for the SAT as well as focus on the marginal students that were the original concern.

In order to determine the effectiveness of the model for predicting GPA, a baseline was used compared to the modelôs results. Since the standard to grade a student's ability has been the SAT score, it was regressed onto GPA by itself. This is the baseline to which the accuracy of the results from the textual analysis was compared to. The prediction errors were compared to the prediction errors using the textual analysis model.

When a student writes their essay, they have enough time to complete the essay, review it and make sure it is representative of their ability. Standardized testing is a high pressure exam that is timed and can potentially be unrepresentative of a student's capabilities. There is a need to find a compliment to this evaluation as well as give students another chance at signaling their ability. Some students may have performance anxiety before standardized testing, partially due to the importance it is given as the main avenue for getting into college. Other students could have been sick or simply just didn't perform well that day.

The main application of this methodology was the marginal student. There will always be high performing students that the university will want and accepting them will be almost automatic. When trying to increase the yield rate, the admissions department needs to focus on marginal students in order to accept those who are more certain to enroll to allow for lost yield rate when courting the high performing students. There is a point in the acceptance process when there is a cluster of students that have similar SAT scores, high school rankings and other standard metrics

achieved in high school. By adding textual analysis to the selection process, the admissions department can further evaluate students with similar metrics without asking for additional information.

First semester GPA was the chosen ability predictor because of its correlation with retention (another concern for the admissions department) and it signals ability to perform at the university level. The essays offer the admissions department a way of seeing a student's ability that may not be apparent in their standardized testing score. The SAT score will still provide a base score in which the decisions will be made, but the essays may provide a way in which the department can separate students at the margin. This means the university can increase retention rates as well as generate better performing students, both of which increase the university's ranking.

Simulating Yield Estimates to Create Confidence Intervals

The current model being used offers a single point estimate on the number of students predicted to enroll each semester. This is made by cumulatively adding all the probabilities. The current model being used is

$$Enrollment = \alpha + \beta_1 academic\_index + \beta_2 high\_school\_percentile + \beta_3 ACT\_BU1 + \beta_4 SATmodel + \beta_5 national\_merit + B_6 foc\_count + \beta_7 scholar\_athlete + \beta_8 res + \beta_9 ans + \beta_{10} app\_span + \beta_{11} mrate + \beta_{12} premiere + \beta_{13} stealth + \beta_{14} self\_init\_cntcts + \beta_{15} religionrate + \beta_{16} christian\_school + \beta_{17} legacy + \beta_{18} web\_app + \beta_{19} bapt + \beta_{20} BUhonors + \beta_{21} hon\_applied + \beta_{22} init\_span + \beta_{23} under100mi + \beta_{24} under50mi + \beta_{25} under200mi + \beta_{26} parents\_alum + \beta_{27} sibling\_alum. + e$$

A yield estimate offers a single estimation, but having a confidence interval offers a more valid expectation of the number of students the university should expect to accept their offer. This methodology is generally called

boosting and provides a range of expectation of attendance with 95% confidence. In order to create the interval, a regression was run using the current admissions model. The returned coefficients and standard errors for each variable in the regression were used to draw a new random coefficient for each variable. These new coefficients were used to predict a yield estimate. This process was run 5,000 times to create a large pool of expected total yield estimates. This range of yield estimates was created to help alleviate randomness intrinsic in the creation of each individual yield estimate.

With 5,000 expected yield estimates, it was possible to calculate a mean and standard deviation to create a confidence interval. However, it cannot be assumed that the yield estimates will take on a normal distribution which would be needed to calculate a standard deviation. Instead, the number of students expected to attend at the 2.5% and 97.5% percentile were used as the end points for the confidence interval.

CHAPTER FIVE

Results

Predicting Probability of Enrollment

Table 5 reports logistic regression results of enrollment for the three models used. The coefficients and error terms are not the primary concern in the methodology of this research. High accuracy is more important than statistical relevance at this point. More work can be done to increase both in the future. It is important to understand the reaction the variable has on the probability. The variables created to signal commitment have a positive effect on probability. The percentage of words that are Baylor specific has the most effect. This effect is understandable since a strong knowledge of Baylor customs and traditions could signal the student is a legacy, visited the campus, or spent time getting to know the university. These examples have proven in the past to be strong indicators of a student's commitment to attending that university.

An intriguing find with the coefficients was that increases in the ability variables dropped a student's probability of enrollment. An explanation of this would be that Baylor receives applications from students at a certain threshold of ability and above. Because of the perceived cost of attending Baylor, students who fall below that threshold may not be applying. As a student's ability increases, they may be applying to more prestigious universities and being accepted there. Baylor is not attracting students who perform lower on the essays, but is losing students who are performing at high levels.

Table 5: Logistic Regression on Enrollment Probability

	(1) R^2	(2) AIC	(3) BIC
commitmentcounter	0.103	0.0440	0.0436
	(4.11)	(5.62)	(5.59)
*interaction*1	1.387	1.344	
	(2.54)	(2.25)	
baylorcounter	0.0705	0.0853	0.0652
	(2.27)	(3.90)	(9.57)
*interaction*2	-0.00175	-0.00207	
	(-2.16)	(-2.91)	
interaction3	-0.00227		
	(-2.13)		
totalbible	0.188		
	(1.85)		
syllables	-0.00493	-0.00512	-0.00480
	(-3.05)	(-3.19)	(-3.08)
interaction4	-0.0599		
	(-1.45)		
admincounter	0.0423	0.0424	0.0426
	(4.46)	(4.47)	(4.50)
*interaction*5	0.00106	0.00114	0.000980
	(1.95)	(2.12)	(1.95)
Essay2Len	-0.000503	-0.000496	
	(-0.91)	(-0.90)	
*interaction*6	-11.89		
	(-2.68)		
baylorpercent	14.81		
	(1.81)		
interaction7	2.974		
	(0.95)		
NumUnique	0.00379	0.00416	0.00418
	(2.58)	(2.86)	(2.98)
faithpercent		-11.23	
		(-2.25)	
GrammarWrong		2.184	
		(0.87)	
Constant	-1.701	-1.704	-1.672
	(-7.30)	(-8.80)	(-18.62)
Observations	7007	7007	7007

t statistics in parentheses

The primary concern of these models was the predictive power they have as a stand-alone model. Table 6 shows the results of the predictions using only the variables created from the textual analysis. Each of the three models performed poorly against actual data. The percentage of correct predictions is inflated because of the lower bias of the predictions and the high number of students not attending that create a large number of true negatives. With only 67% correctly predicted, the model does not provide significant results. These results are not adequate to allow any of them to work as a stand-alone model to predict probability of enrollment. The low false positive rates show that the model has potential benefits. In order to increase the yield rate, the false positive rate needs to be low. These numbers could mean that the variables used in these models could be helpful in the overall model currently being used. With the high effectiveness rates of the current model, there may only be a slight increase using the textual analysis, but it would help to remove some uncertainty.

Finding Hidden Ability

This model represents the linear effect each of the textual analysis variables has on the dependent variable, the student's first semester GPA. Table 7 displays the results for each of the subgroups of SAT scores. The tails both suffered from a low number of observations. This data only represents students who enrolled, finished the first semester and submitted the optional essays on their application. Either making the essays mandatory or creating a benefit for submission could increase the number of students participating and offer more observations to make a better fit for the model. Each group had different coefficients for the variables and some varied greatly, even switching from a positive relationship to a negative. This shows the

importance of using this model with the subgroups. Prediction of marginal students was able to be made with more accuracy using these sub groupings.

Table 6: Enrollment Predictions

Measuring Decision	Observations	Percent Correct	False Positive Percentage
AIC	24453	67.23%	14.77%
R^2	24453	67.75%	12.69%
BIC	24453	67.79%	12.71%

Table 8 shows the predicted GPA for each subgroup as a whole. For robustness, SAT scores were regressed onto GPA by themselves and used in the prediction model. Since SAT scores have been the standard for predicting a student's ability, it served as the best benchmark to the model using textual analysis. The predictions using textual analysis were all higher than the actual GPA. The SAT score based predictions were almost identical to the actual GPA for each subgroup. These results show that the SAT should still be used as part of the predicting method, with textual analysis used to compliment it. New stepwise regression can be done to find a model that includes high school metrics, standardized testing results and textual analysis.

Table 7: OLS Regression Results

	SAT<1050	1050 < SAT < 1150	1150 < SAT < 1250	1250 < SAT< 1350	1350 < SAT< 1450	SAT >1450	Population
Syllables	0.00187	-0.00849	-0.0100	-0.00775	-0.0406	-0.00614	-0.0125
	(0.09)	(-0.66)	(-1.09)	(-0.75)	(-2.53)	(-0.53)	(-2.46)
*NumUnique * lexrich*	-0.00181	0.0130	0.0183	0.00882	0.0536	0.0114	0.0201
	(-0.06)	(0.67)	(1.38)	(0.60)	(2.21)	(0.63)	(2.75)
*fleschreading * GrammarWrong*	2.263	0.356	0.367	-0.238	0.750	1.218	0.342
	(0.97)	(0.73)	(0.80)	(-0.42)	(0.64)	(0.67)	(1.73)
$Sentences^2$	0.000221	-0.000417	0.000551	-0.000539	0.000646	0.000305	0.000186
	(0.15)	(-0.45)	(1.83)	(-1.17)	(0.92)	(0.63)	(1.00)
$TotalLen^2$	0.00000620	-0.00000400	-0.0000152	-0.0000115	-0.0000426	-0.00000602	-0.0000139
	(0.26)	(-0.25)	(-1.36)	(-0.98)	(-2.36)	(-0.44)	(-2.29)
GrammarWrong	-331.1	-31.92	-37.64	70.58	-224.4	-72.93	-33.58
	(-1.11)	(-0.72)	(-0.81)	(0.80)	(-1.64)	(-0.26)	(-1.68)
*syllables * TotalLen*	-0.00000363	0.00000249	0.0000100	0.00000899	0.0000334	0.00000329	0.00000935
	(-0.21)	(0.22)	(1.23)	(1.04)	(2.57)	(0.34)	(2.09)
Sentences	-0.0254	-0.0444	-0.0699	0.0868	-0.0721	-0.0325	-0.0778
	(-0.13)	(-0.27)	(-0.83)	(0.84)	(-0.48)	(-0.36)	(-1.99)
$Fleschreading^3$	-0.00000272	-0.00000277	-0.00000228	-0.000000849	-0.0000103	-0.00000322	-0.00000306
	(-0.40)	(-1.01)	(-0.97)	(-0.32)	(-2.26)	(-0.72)	(-2.72)
*fleschreading * fleschgrade*	-0.00769	-0.00240	-0.00452	0.000845	-0.0132	-0.00320	-0.00381
	(-0.69)	(-0.62)	(-1.62)	(0.21)	(-1.97)	(-0.44)	(-2.72)
*fleschreading * fleschgrade*GrammarWrong*	0.233	0.0245	0.0188	-0.0772	0.204	-0.00446	0.0199
	(1.08)	(0.85)	(0.76)	(-0.87)	(1.25)	(-0.01)	(1.85)
*sentences * WordLength*	-0.0110	0.0141	-0.00205	-0.00602	-0.00289	-0.000342	0.00996
	(-0.36)	(0.56)	(-0.12)	(-0.36)	(-0.11)	(-0.03)	(1.70)
Constant	10.16	4.521	6.381	2.646	17.58	6.381	6.077
	(1.07)	(1.48)	(2.54)	(0.76)	(3.44)	(1.03)	(4.95)
Observations	50	296	391	347	126	43	1253

t statistics in parentheses

The population averages were all relatively the same, so additional error testing was done. Table 9 shows the root means square prediction error (RMSPE) for the textual analysis, the SAT and a randomly generated number assigned to each student between 0 and 1. The error rate of the random number generated was checked to see if the randomness could show relevance. This could discredit the model since a random number assigned to each student should not be a strong predictor of their GPA. The RMSPE was used because it squares the error in order to make all numbers positive for comparison. If this wasn't done, a predicted GPA below the actual GPA for one student could average out to zero potentially with someone who had a predicted average that was higher than the actual GPA. This would make the error rates seem better than they actually are. The error rate for the textual analysis was lower than both the randomly generated number as well as the SAT error rate. This shows that the combination of variables used from textual analysis could offer another avenue in determining which students at the margin should be accepted into Baylor.

Table 8: Predicted GPAs

Rank	Predicted GPA With Textual Analysis	Predicted GPA With SAT	Actual GPA
1	3.023802	3.014625	2.930003
2	2.859733	2.816324	2.841327
3	3.141526	3.078084	3.082979
4	3.39128	3.309947	3.313185
5	3.552784	3.506608	3.51439
6	4.044257	3.698329	3.686065
Total	3.455361	3.150734	3.158706

Simulating Yield Estimates to Create Confidence Intervals

The new confidence interval creation was calculated offer a range in which the student yield could be expected to fall within. A single point estimate does not allow

for uncertainty in the model, and the confidence intervals lessen this problem since uncertainty is always inherent in predictions. The simulations were run several different ways. All combinations of training and testing were used to see how close yield estimates were to the actual enrollment. There were combinations using two of the years to train and one to test. Another training set included a random half of all the years and tested on the other half. There were 5,000 expected yields used to model the uncertainty. A 95% confidence interval was created by finding the expected yield at both the 2.5 percentile and the 97.5 percentile.

Table 10 shows the results of finding the confidence intervals using simulation. The results become more accurate as past data was used to predict current data. When predicting 2013, the resulting point estimate was only two students away from being identical to the actual data. When using future and past data to predict 2012, a very high estimate was returned. Extremely low results were returned when predicting 2011 using data posted in future years. This is in line with the results from Van Dyke's research.

Table 9: Root Mean Square Prediction Errors

Variable	Observations	Mean
Textual Analysis	55081	0.6940738
SAT	55081	0.7474648
Random Number	55081	0.7861156

Table 10: Simulation Results

Data Predicted On	Low End	High End	Point Estimate	Actual Enrollment
2013 using 2011 &	3172	3205	3190	3188
2012 using 2011 &	3612	3650	3631	3254
2011 using 2012 &	1214	1239	1226	3033
Using half to predict	3275	3318	3296	4793

CHAPTER SIX

Discussions

The textual analysis used in this methodology was not strong enough to stand-alone as a decision rule. It can be used in conjunction with more comprehensive models to help increase their predictive power. When adding the created variables to the full data set that includes information gathered during the application process, which includes SAT score, high school rank, etc. and running the stepwise regressions, some of the new variables were selected as good fits for a model. More data cleaning needs to be completed due to the unique identifiers used to retain anonymity do not match up every time. This would need to be adjusted to have a more complete data set. A model has been submitted for use with the admissions department, and the textual analysis variables could be used for the Fall 2014 incoming freshmen.

An area that needs additional development is finding better signals within the essays to help improve the performance of the model. The variables created were from personal reading of the essays to gather ideas. More readers involved could produce more ideas or find correlations in essays by students who enroll. Voluntary submission of the essays presents a problem that could easily be remedied by making them mandatory. There could also be selection bias built into those who choose to submit the essays. The participation rates show no true correlation with students who actually enroll however there are many students that submit them and are not accepted.

The essay questions could be altered to ask about the student's motivation for attending Baylor. The variables created that help define signals in the essays were geared towards finding a student's motivation. If the essay focused on this, the student would be encouraged to use Baylor specific words or expressions of their faith. Being a large private university, Baylor has to compete not only with public state universities like Texas A&M and the UT, but also with other private institutions like TCU and SMU. Those two set of schools draw from different groups of students, with Baylor in the middle. Being able to find people from both sets was the goal of this research, and asking the right questions is the key to accomplishing this.

The first semester GPA used was the most accurate information available to make predictions with. The research would be better fitted if core classes could be accessed to find ability. The first semester between two students could vary widely depending on the classes they take. By focusing on core freshman classes, such as calculus or literature, a more accurate model could be made. However, the number of observations that would be available would probably suffer because the data set is being limited even more. The number of enrolled students that complete the essays needs to be increased to improve results.

By creating a large number of yield estimates and calculating a confidence interval from them, the results appear to be more accurate than running a single regression for the 2013 sample using chronological data. However, the results for 2011 and 2012 were not accurate enough to be reliable, which has been the problem with previous results as well. (Van Dyke, 2014) My conclusion is that preferences for college of enrollment are a forward progressing set of choices. The markets and preferences change yearly. Some universities become more prestigious over time and others less. The fact that results from chronological data is much more accurate than

data in reverse chronological order suggests that these changes build upon one another and need to be taken into consideration. The simple solution for this problem is to acquire more data from previous years. However, new data to be collected is added periodically and the models have to be re-evaluated and changed. A moving model updated yearly that includes the last three to four years may be the needed application of this research. I recommend using the simulation method that incorporates previous years to create a range of expected values for each student in order to reduce the uncertainty of having a single point estimate.

APPENDICES

APPENDIX A

Summary Statistics

Table 11: Enrollment Regression Summary Statistics

Method	# Observations	Mean	Std. Dev.	Min	Max
R^2	13971	.3137741	.0925661	.0163004	.9997028
AIC	13971	.3177392	.0982734	.0603836	.9999732
BIC	13971	.3177145	.0953883	.1345734	.9618691

Table 12: Ability Regression Summary Statistics

Model	# Observations	Mean	Std. Dev.	Min	Max
AIC	9524	3.229942	.326763	1.809234	4.19948
SAT	48879	3.166352	.2463845	2.0242	3.93375
Random Number	48884	3.16257	.0105589	3.144304	3.18086

APPENDIX B

Python Code ï Variable Generating

```
from __future__ import division
import nltk
import csv
import re
from string import punctuation
import enchant
from enchant.checker import SpellChecker
faithwords = ['shall', 'unto', 'lord', 'thou', 'thy', 'ye', 'god', 'son', 'hath', 'israel', 'king',
'people', 'house', 'before', 'children', 'against', 'shalt', 'land', 'day', 'hand', 'behold', 'saith',
'sons', 'hast', 'o', 'over', 'she', 'david', 'great', 'jesus', 'thine', 'father', 'neither', 'give',
'take', 'am', 'forth', 'brought', 'name', 'away', 'pass', 'two', 'according', 'days', 'city',
'earth', 'moses', 'thereof', 'whom', 'know']
commitmentwords = ['pride', 'visit', 'leader', 'unity', 'campus', 'legacy', 'important',
'environment', 'only', 'home', 'comfort', 'mom', 'dad', 'mother', 'father', 'family']
baylorwords = ['baylor', 'bears', 'bear', 'waco', 'oso', 'quadrangle', 'ivy', 'hometown',
'burleson', 'hankamer', 'louise', 'herrington', 'truett']
oldbible = ['genesis', 'exodus', 'leviticus', 'numbers', 'deuteronomy', 'joshua', 'judges',
'ruth', 'samuel', 'kings', 'chronicles', 'ezra', 'nehemiah', 'esther', 'job', 'psalm', 'proverbs',
'ecclesiastes', 'solomon', 'isaiah', 'jermiah', 'lamentations', 'ezekiel', 'daniel', 'hosea',
'joel', 'amos', 'obadiah', 'jonah', 'micah', 'nahum', 'habakkuk', 'zephaniah', 'haggai',
'zechariah', 'malachi']
newbible = ['matthew', 'mark', 'luke', 'john', 'acts', 'romans', 'corinthians', 'galatians',
'ephesians', 'philippians', 'thessalonians', 'timothy', 'titus', 'philemon', 'hebrews', 'james',
'peter', 'jude', 'revelation']
adminwords = ['top', 'choice', 'first', 'dream', 'dreamed', 'always', 'growing', 'grown',
'grow', 'wonderful', 'visit', 'blessed', 'experience', 'passion', 'passionate', 'tradition',
'atmosphere', 'values', 'perfect', 'heart']

dictionary = enchant.Dict("en_US")
chkr = SpellChecker("en_US")

with open('additionalstatement.csv', 'rb') as csvfile:
data = csv.reader(csvfile, delimiter=",")
writer = csv.writer(open('additionaloutput.csv', 'wb'))

for row in data:
faithcounter = 0
grammarcounter = 0
commitmentcounter = 0
baylorcounter = 0
oldcounter = 0
newcounter = 0
```

```
admincounter = 0
count = 0
vowels = 'aeiouy'
sentences = 0

#######Formating data
row3 = row[4]
row3 = row3.lower().replace(' ', ' ')
row4 = row[5]
row4 = row4.lower().replace(' ', ' ')
row3 = row3.replace('\n', '')
row4 = row4.replace('\n', '')
for line in row3:
sentences += line.count('.')+ line.count ('!')+ line.count('?')

for line in row4:
sentences += line.count('.')+ line.count ('!')+ line.count('?')
for p in list(punctuation):
row3 = row3.replace(p, '')
row4 = row4.replace(p, '')
whybaylor = re.split(' ', row3)
lookingfor = re.split(' ', row4)

entire = whybaylor + lookingfor
#######Language Numbers
#Total Number of Words
whybaylorlen = len(whybaylor)
lookingforlen = len(lookingfor)

#Total Number of Different Words
numunique = len(set(entire))

#Number of Misspelled Words
chkr.set_text(row)
for word in row:
grammarcounter = grammarcounter + 1
#Lexical Richness or How many times each word was used
lexrich = len(entire) / len(set(entire))
#Calculations
total_length = lookingforlen + whybaylorlen
word_length = (sum(len(word) for word in whybaylor)+sum(len(word) for word in
lookingfor))/(len(whybaylor)+len(lookingfor))
grammar_wrong = grammarcounter / total_length
#######Word Counters
for word in whybaylor:
if word in faithwords:
faithcounter = faithcounter + 1
for word in lookingfor:
if word in faithwords:
faithcounter = faithcounter + 1
```

```
for word in whybaylor:
if word in commitmentwords:
commitmentcounter = commitmentcounter + 1
for word in lookingfor:
if word in commitmentwords:
commitmentcounter = commitmentcounter + 1
for word in whybaylor:
if word in baylorwords:
baylorcounter = baylorcounter + 1
for word in lookingfor:
if word in baylorwords:
baylorcounter =baylorcounter + 1

for word in whybaylor:
if word in oldbible:
oldcounter = oldcounter + 1
for word in lookingfor:
if word in oldbible:
oldcounter =oldcounter + 1

for word in whybaylor:
if word in newbible:
newcounter = newcounter + 1
for word in lookingfor:
if word in newbible:
newcounter = newcounter + 1

for word in whybaylor:
if word in adminwords:
admincounter = admincounter + 1
for word in lookingfor:
if word in adminwords:
admincounter = admincounter + 1
for word in whybaylor:
for index in range(0,len(word)):
if word[index] in vowels and word[index-1] not in vowels:
count +=1
if word.endswith('e'):
count -= 1
if word.endswith('le'):
count+=1
if count == 0:
count +=1
for word in lookingfor:
for index in range(0,len(word)):
if word[index] in vowels and word[index-1] not in vowels:
count +=1
if word.endswith('e'):
count -= 1
if word.endswith('le'):
```

```
count+=1
if count == 0:
count +=1
commitment_percentage = commitmentcounter / total_length
faith_percentage = faithcounter / total_length
baylor_percentage = baylorcounter / total_length
totalbible = newcounter + oldcounter
bible_percentage = totalbible / total_length
admin_percentage = admincounter / total_length
if sentences == 0:
sentences = 1
var1 = total_length / sentences
var2 = count / total_length
readingease = 206.835 - 1.015 * var1 - 84.6 * var2
gradelevel = 0.39 * var1 + 11.8 * var2 -15.59
#######Outputting new variables
#Quality of writing variables
row.append(whybaylorlen) #Number of Words in Essay1
row.append(lookingforlen) #Number of Words in Essay2
row.append(total_length) #Total Number of Words in Both Essays Combined
row.append(grammar_wrong) #Percent of Misspelled Words in Both Essays
row.append(word_length) #Average Number of Letters in a Word
row.append(numunique) #Number of Unique Words in Both Essays Combined
row.append(lexrich) #Average Number of Times Each Word was Used
#Signaling variables
row.append(commitmentcounter)
row.append(commitment_percentage)
row.append(faithcounter)
row.append(faith_percentage)
row.append(baylorcounter)
row.append(baylor_percentage)
row.append(totalbible)
row.append(bible_percentage)
row.append(admincounter)
row.append(admin_percentage)
#Flesch-Kincaid Metrics
row.append(count)
row.append(sentences)
row.append(readingease)
row.append(gradelevel)
writer.writerow(row)
```

APPENDIX C

Python Code ï Bible Input

```
from collections import Counter
import csv
import re
import nltk
import enchant
from enchant.checker import SpellChecker
from string import punctuation

dictionary = enchant.Dict("en_US")
chkr = SpellChecker("en_US")

with open('thebible.txt', 'r') as thebible:
biblewords = thebible.read().replace('\n', '')
biblewords = biblewords.lower().replace(' ', ' ')
for p in list(punctuation):
biblewords = biblewords.replace(p, '')
biblesplit = re.split(' ', biblewords)

def remove_values_from_list(the_list, val):
return [value for value in the_list if value != val]

biblesplit = remove_values_from_list(biblesplit, 'the')
biblesplit = remove_values_from_list(biblesplit, 'and')
biblesplit = remove_values_from_list(biblesplit, 'of')
biblesplit = remove_values_from_list(biblesplit, 'to')
biblesplit = remove_values_from_list(biblesplit, 'that')
biblesplit = remove_values_from_list(biblesplit, 'in')
biblesplit = remove_values_from_list(biblesplit, 'he')
biblesplit = remove_values_from_list(biblesplit, 'for')
biblesplit = remove_values_from_list(biblesplit, 'i')
biblesplit = remove_values_from_list(biblesplit, 'his')
biblesplit = remove_values_from_list(biblesplit, 'a')
biblesplit = remove_values_from_list(biblesplit, 'they')
biblesplit = remove_values_from_list(biblesplit, 'upon')
biblesplit = remove_values_from_list(biblesplit, 'will')
biblesplit = remove_values_from_list(biblesplit, 'from')
biblesplit = remove_values_from_list(biblesplit, 'upon')
biblesplit = remove_values_from_list(biblesplit, 'as')
biblesplit = remove_values_from_list(biblesplit, 'thee')
biblesplit = remove_values_from_list(biblesplit, 'be')
biblesplit = remove_values_from_list(biblesplit, 'is')
biblesplit = remove_values_from_list(biblesplit, 'not')
```

```
biblesplit = remove_values_from_list(biblesplit, 'with')
biblesplit = remove_values_from_list(biblesplit, 'him')
biblesplit = remove_values_from_list(biblesplit, 'them')
biblesplit = remove_values_from_list(biblesplit, 'it')
biblesplit = remove_values_from_list(biblesplit, 'all')
biblesplit = remove_values_from_list(biblesplit, 'have')
biblesplit = remove_values_from_list(biblesplit, 'me')
biblesplit = remove_values_from_list(biblesplit, 'was')
biblesplit = remove_values_from_list(biblesplit, 'which')
biblesplit = remove_values_from_list(biblesplit, 'my')
biblesplit = remove_values_from_list(biblesplit, 'but')
biblesplit = remove_values_from_list(biblesplit, 'said')
biblesplit = remove_values_from_list(biblesplit, 'are')
biblesplit = remove_values_from_list(biblesplit, 'their')
biblesplit = remove_values_from_list(biblesplit, 'this')
biblesplit = remove_values_from_list(biblesplit, 'had')
biblesplit = remove_values_from_list(biblesplit, 'into')
biblesplit = remove_values_from_list(biblesplit, 'on')
biblesplit = remove_values_from_list(biblesplit, 'by')
biblesplit = remove_values_from_list(biblesplit, 'at')
biblesplit = remove_values_from_list(biblesplit, 'let')
biblesplit = remove_values_from_list(biblesplit, 'go')
biblesplit = remove_values_from_list(biblesplit, 'went')
biblesplit = remove_values_from_list(biblesplit, 'come')
biblesplit = remove_values_from_list(biblesplit, 'if')
biblesplit = remove_values_from_list(biblesplit, 'when')
biblesplit = remove_values_from_list(biblesplit, 'were')
biblesplit = remove_values_from_list(biblesplit, 'out')
biblesplit = remove_values_from_list(biblesplit, 'man')
biblesplit = remove_values_from_list(biblesplit, 'you')
biblesplit = remove_values_from_list(biblesplit, 'up')
biblesplit = remove_values_from_list(biblesplit, 'there')
biblesplit = remove_values_from_list(biblesplit, 'then')
biblesplit = remove_values_from_list(biblesplit, 'came')
biblesplit = remove_values_from_list(biblesplit, 'one')
biblesplit = remove_values_from_list(biblesplit, 'we')
biblesplit = remove_values_from_list(biblesplit, 'your')
biblesplit = remove_values_from_list(biblesplit, 'also')
biblesplit = remove_values_from_list(biblesplit, 'an')
biblesplit = remove_values_from_list(biblesplit, 'so')
biblesplit = remove_values_from_list(biblesplit, 'no')
biblesplit = remove_values_from_list(biblesplit, 'even')
biblesplit = remove_values_from_list(biblesplit, 'now')
biblesplit = remove_values_from_list(biblesplit, 'do')
biblesplit = remove_values_from_list(biblesplit, 'us')
biblesplit = remove_values_from_list(biblesplit, 'therefore')
biblesplit = remove_values_from_list(biblesplit, 'these')
biblesplit = remove_values_from_list(biblesplit, 'because')
biblesplit = remove_values_from_list(biblesplit, 'her')
biblesplit = remove_values_from_list(biblesplit, 'men')
```

```
biblesplit = remove_values_from_list(biblesplit, 'saying')
biblesplit = remove_values_from_list(biblesplit, 'made')
biblesplit = remove_values_from_list(biblesplit, 'every')
biblesplit = remove_values_from_list(biblesplit, 'after')
biblesplit = remove_values_from_list(biblesplit, 'our')
biblesplit = remove_values_from_list(biblesplit, 'or')
biblesplit = remove_values_from_list(biblesplit, 'down')
biblesplit = remove_values_from_list(biblesplit, 'things')
biblesplit = remove_values_from_list(biblesplit, 'make')
biblesplit = remove_values_from_list(biblesplit, 'put')
biblesplit = remove_values_from_list(biblesplit, 'among')
biblesplit = remove_values_from_list(biblesplit, 'who')
biblesplit = remove_values_from_list(biblesplit, 'may')
biblesplit = remove_values_from_list(biblesplit, 'our')
biblesplit = remove_values_from_list(biblesplit, 'or')
biblesplit = remove_values_from_list(biblesplit, 'did')
biblesplit = remove_values_from_list(biblesplit, 'any')
biblesplit = remove_values_from_list(biblesplit, 'what')
biblesplit = remove_values_from_list(biblesplit, 'say')
biblesplit = remove_values_from_list(biblesplit, 'should')
fdist1 = nltk.FreqDist(biblesplit)
vocabulary1 = fdist1.keys()
output = vocabulary1[:50]
print output
```

APPENDIX D

Python Code ï Traditions Retrieval

```
from __future__ import division
import nltk
import re
from string import punctuation
import enchant
from enchant.checker import SpellChecker
from collections import Counter

dictionary = enchant.Dict("en_US")
chkr = SpellChecker("en_US")

with open('baylortraditions.txt', 'r') as baylortraditions:
baylorwords = baylortraditions.read().replace('\n', '')

baylorwords = baylorwords.lower().replace(' ', ' ')
for p in list(punctuation):
baylorwords = baylorwords.replace(p, '')
baylorsplit = re.split(' ', baylorwords)

with open('uttraditions.txt', 'r') as uttraditions:
utwords = uttraditions.read().replace('\n', '')

utwords = utwords.lower().replace(' ', ' ')
for p in list(punctuation):
utwords = utwords.replace(p, '')
utsplit = re.split(' ', utwords)
with open('aggietraditions.txt', 'r') as aggietraditions:
aggiewords = aggietraditions.read().replace('\n', '')
aggiewords = aggiewords.lower().replace(' ', ' ')
for p in list(punctuation):
aggiewords = aggiewords.replace(p, '')
aggiesplit = re.split(' ', aggiewords)
with open('smutraditions.txt', 'r') as smutraditions:
smuwords = smutraditions.read().replace('\n', '')
smuwords = smuwords.lower().replace(' ', ' ')
for p in list(punctuation):
smuwords = smuwords.replace(p, '')
smusplit = re.split(' ', smuwords)

c =Counter(smusplit)
for word in set(baylorsplit):
print word, c.get(word, 0)
```

APPENDIX E

Python Code ï Essay Breakdown

```
from collections import Counter
import csv
import re
import nltk
from string import punctuation
from nltk.corpus import stopwords

enrollwords = []
noenrollwords = []

with open('2011ShortAnswers.csv', 'rb') as csvfile:
data = csv.reader(csvfile, delimiter=",")
writer = csv.writer(open('keyword.csv', 'wb'))

for row in data:

#Format Data
row3 = row[3]
row3 = row3.lower().replace(' ', ' ')
row4 = row[4]
row4 = row4.lower().replace(' ', ' ')
row3 = row3
for p in list(punctuation):
row3 = row3.replace(p, '')
row4 = row4.replace(p, '')
whybaylor = re.split(' ', row3)
lookingfor = re.split(' ', row4)
allrow = row3 + row4
entire = whybaylor + lookingfor
#Removing words that are of no importance
def remove_values_from_list(the_list, val):
return [value for value in the_list if value != val]

entire = remove_values_from_list(entire, 'a')
entire = remove_values_from_list(entire, 'i')
entire = remove_values_from_list(entire, 'my')
entire = remove_values_from_list(entire, 'me')
entire = remove_values_from_list(entire, 'to')
entire = remove_values_from_list(entire, 'and')
entire = remove_values_from_list(entire, 'was')
entire = remove_values_from_list(entire, 'as')
entire = remove_values_from_list(entire, 'you')
```

```
entire = remove_values_from_list(entire, 'just')
entire = remove_values_from_list(entire, 'so')
entire = remove_values_from_list(entire, 'will')
entire = remove_values_from_list(entire, 'also')
entire = remove_values_from_list(entire, 'do')
entire = remove_values_from_list(entire, 'for')
entire = remove_values_from_list(entire, 'of')
entire = remove_values_from_list(entire, 'it')
entire = remove_values_from_list(entire, 'the')
entire = remove_values_from_list(entire, 'in')
entire = remove_values_from_list(entire, 'is')
entire = remove_values_from_list(entire, 'at')
entire = remove_values_from_list(entire, 'am')
entire = remove_values_from_list(entire, 'get')
entire = remove_values_from_list(entire, 'has')
entire = remove_values_from_list(entire, 'can')
entire = remove_values_from_list(entire, 'with')
entire = remove_values_from_list(entire, 'all')
entire = remove_values_from_list(entire, 'an')
entire = remove_values_from_list(entire, 'be')
entire = remove_values_from_list(entire, 'that')
entire = remove_values_from_list(entire, 'on')
entire = remove_values_from_list(entire, 'not')
entire = remove_values_from_list(entire, 'want')
entire = remove_values_from_list(entire, 'would')
entire = remove_values_from_list(entire, 'are')
entire = remove_values_from_list(entire, 'have')
entire = remove_values_from_list(entire, 'but')
entire = remove_values_from_list(entire, 'about')
entire = remove_values_from_list(entire, 'like')
entire = remove_values_from_list(entire, 'from')
entire = remove_values_from_list(entire, 'because')
entire = remove_values_from_list(entire, 'this')
entire = remove_values_from_list(entire, 'by')
entire = remove_values_from_list(entire, '')
entire = remove_values_from_list(entire, 'many')
entire = remove_values_from_list(entire, 'able')
entire = remove_values_from_list(entire, 'where')
entire = remove_values_from_list(entire, 'being')
entire = remove_values_from_list(entire, 'what')
entire = remove_values_from_list(entire, 'been')

entire = remove_values_from_list(entire, 'one')
entire = remove_values_from_list(entire, 'myself')
entire = remove_values_from_list(entire, 'when')
entire = remove_values_from_list(entire, 'well')
entire = remove_values_from_list(entire, 'very')
entire = remove_values_from_list(entire, 'more')
entire = remove_values_from_list(entire, 'there')
entire = remove_values_from_list(entire, 'their')
```

```
entire = remove_values_from_list(entire, 'who')
entire = remove_values_from_list(entire, 'or')
entire = remove_values_from_list(entire, 'other')
entire = remove_values_from_list(entire, 'make')

#Creating a list of the top words used by individuals
fdist1 = nltk.FreqDist(entire)
output = fdist1.keys()
# output = vocabulary1[:5]
# output2 = sorted([w for w in set(entire) if len(w) > 4 and fdist1[w] > 2])
# output3 = fdist1.hapaxes()
if row[1] == '1':
enrollwords = enrollwords + output
if row[1] == '0':
noenrollwords = noenrollwords + output
#Taking the lists of top words used by individuals and creating top used list in
aggregate
overallyes = nltk.FreqDist(enrollwords)
yesvocab = overallyes.keys()
yeswords = yesvocab[:35]

overallno = nltk.FreqDist(noenrollwords)
novocab = overallno.keys()
nowords = novocab[:35]

print yeswords
print nowords
```

APPENDIX F

STATA Code

```
/***********************************************/
/* Stata program for Admissions Research       */
/* Van Pham, and Stephen Ryan Beckham (RA)     */
/* last revised by SRB Apr 3 2014              */
/***********************************************/

capture program drop _all

program admissions

        clear
        set more off

        /*********************************************************/
        /* Data mining done in Python.  Output files are         */
        /* generated there and then moved to STATA for           */
        /* statistical work.  Below set is for cleaning and      */
        /* merging data.                                         */
        /*********************************************************/

         *Python
         *Clean
         *appendnmerge
         *changengenerate

        /*********************************************************/
        /* The code to predict propensity to enroll              */
        /*********************************************************/

        *vselectpropensity
        *predictpropensity

        /*********************************************************/
        /* The code to predict ability                           */
        /*********************************************************/

        *vselectgpa
        *text2gpa
        *vselectsat
        *text2sat
        *sat2gpa
        *hsp2gpa
```

```
*text2gpamarginal
        *textsat2gpa
        *textsathsp2gpa

        /*********************************************************/
        /* The code is using the open answer data set          */
        /*********************************************************/

        *openanswerconvert

        /*********************************************************/
        /* The code to predict retention                       */
        /*********************************************************/

        *vselectretention
        *predictretention

        *subgroupingsat
        *fregressionlevels


end

/*********************************/

program Python

        //Pulling Python Files
        insheet using 2011output.csv
        save 2011output.dta
        clear
        insheet using 2012output.csv
        save 2012output.dta
        clear
        insheet using 2013output.csv
        save 2013output.dta
        clear

end //-------------Python----------------


/*********************************/

program Clean

        //Cleaning File that will allow the merge of crosswalk and random_ID
        import excel "C:\Documents and Settings\stephen_beckham\My
Documents\Match Crosswalk.xls", sheet("Matched")
        rename A crosswalk
        rename B random_ID
```

```
        rename C Term
        drop if crosswalk == "CROSSWALK"
        destring crosswalk, replace
        destring random_ID, replace
        destring Term, replace
        save matchcrosswalk.dta
        clear

end //-------------Clean----------------

/*******************************/

program appendnmerge

        //Appending all the Python datasets into one data set
        use 2011output.dta
        append using 2012output.dta
        append using 2013output.dta
        save combinedstata.dta
        clear

        //Merging the data set created in Python with Sarah's data set
        use combinedstata.dta
        merge m:m crosswalk using matchcrosswalk.dta, nogen
        merge m:1 random_ID using sarahsdta.dta, nogen
        save masterlist.dta
        clear

end //-------------appendnmerge----------------

/*******************************/

program changengenerate

        //Cleaning up the variables into memorable names
        use masterlist.dta
        rename baylor_cont_text WhyBaylor
        rename univ_look_text Lookingfor
        rename v7 Essay1Len
        rename v8 Essay2Len
        rename v9 TotalLen
        rename v10 GrammerWrong
        rename v11 WordLength
        rename v12 NumUnique
        rename v13 lexrich
        rename v14 commitmentcounter
        rename v15 commitmentpercent
        rename v16 faithcounter
        rename v17 faithpercent
        rename v18 baylorcounter
```

```
rename v19 baylorpercent
rename v20 totalbible
rename v21 biblepercent
rename v22 admincounter
rename v23 adminpercent
rename v24 syllables
rename v25 sentences
rename v26 fleschreading
rename v27 fleschgrade

//Creaing variables to assist with vselect
gen newrandom = runiform()
gen var1 = biblepercent * faithcounter
gen var2 = totalbible * totalbible
gen var3 = baylorcounter * baylorcounter
gen var4 = faithcounter * faithcounter
gen var5 = commitmentcounter * commitmentcounter
gen var6 = faithpercent * totalbible
gen var7 = baylorpercent * commitmentcounter
gen var8 = commitmentpercent * baylorcounter
gen var9 = commitmentpercent + baylorpercent
gen var10 = biblepercent + faithpercent
gen var11 = NumUnique * lexrich
gen var12 = TotalLen * WordLength
gen var13 = lexrich * GrammerWrong
gen var14 = biblepercent + baylorpercent + faithpercent + commitmentpercent
gen var15 = GrammerWrong + commitmentpercent
gen var16 = biblepercent * commitmentcounter
gen var17 = totalbible * commitmentpercent
gen var18 = faithcounter * baylorpercent


gen gpa2 = TotalLen * TotalLen
gen gpa3 = WordLength * WordLength
gen gpa4 = NumUnique * NumUnique
gen gpa5 = lexrich * lexrich
gen gpa6 = syllables * syllables
gen gpa7 = sentences * sentences
gen gpa8 = fleschreading * fleschreading
gen gpa9 = fleschgrade * fleschgrade
gen gpa10 = fleschreading * fleschgrade
gen gpa11 = TotalLen * WordLength * sentences
gen gpa12 = fleschreading * GrammerWrong
gen gpa13 = fleschgrade * GrammerWrong
gen gpa14 = fleschreading * fleschgrade * GrammerWrong
gen gpa15 = NumUnique * lexrich
gen gpa16 = syllables * TotalLen
gen gpa17 = sentences * WordLength
gen gpa18 = lexrich * GrammerWrong
gen gpa19 = fleschreading * fleschreading * fleschreading
```

```
        gen gpa20 = fleschgrade * fleschgrade * fleschgrade

        save masterlist.dta, replace
        clear

end //-------------changengenerate----------------

/*******************************/

program vselectpropensity

        use masterlist.dta

        //Those that were not admitted are not being used
        drop if admitted == 0

        //The vselect will be fixed on a single year, this will need to be changed for
robustness checks
        vselect registered Essay1Len Essay2Len TotalLen GrammerWrong
WordLength NumUnique lexrich commitmentpercent commitmentcounter
faithcounter faithpercent baylorcounter baylorpercent totalbible biblepercent
admincounter adminpercent syllables sentences fleschreading fleschgrade var1 var2
var3 var4 var5 var6 var7 var8 var9 var10 var11 var12 var13 var14 var15 var16 var17
var18 if newrandom > .5, best


end //-------------veselectpropensity----------------

/*******************************/

program predictpropensity

        clear

        use f:\Users\Stephen_Beckham\masterlist.dta

        //Those that were not admitted are not being used
        drop if admitted == 0

        //Logits run based on the vselect recommendations.  These will need to be
changed for robustness check
        quiet logit registered commitmentcounter var18 baylorcounter var3 var5
totalbible syllables var2 admincounter var12 Essay2Len var14 baylorpercent var15
NumUnique if newrandom > .7
        quiet predict pr_enroll_r2, pr
        quiet logit registered commitmentcounter var18 baylorcounter var3 syllables
admincounter var12 Essay2Len faithpercent GrammerWrong NumUnique if
newrandom > .5
        quiet predict pr_enroll_aic, pr
        quiet logit registered commitmentcounter baylorcounter syllables
```

```
admincounter var12 NumUnique if newrandom > .5
        quiet predict pr_enroll_bic, pr
        quiet logit registered Essay1Len Essay2Len TotalLen GrammerWrong
WordLength NumUnique lexrich commitmentpercent commitmentcounter
faithcounter faithpercent baylorcounter baylorpercent totalbible biblepercent
admincounter adminpercent var1 var2 var3 var4 var5 var6 var7 var8 var9 var10 var11
var12 var13 var14 var15 var16 var17 var18 if newrandom > .5
        quiet predict pr_enroll_all, pr

        //Droping term used in regressions to get out of sample predictions
        drop if newrandom > .5

        //vselect's choice of R^2
        gen enrollhatr2 = 0
        replace enrollhatr2 = 1 if pr_enroll_r2 > 0.4
        gen falseposr2 = 0
        replace falseposr2 = 1 if registered == 0 & enrollhatr2 == 1
        gen falsenegr2 = 0
        replace falsenegr2 = 1 if registered == 1 & enrollhatr2 == 0
        gen trueposr2 = 0
        replace trueposr2 = 1 if registered == 1 & enrollhatr2 == 1
        gen truenegr2 = 0
        replace truenegr2 = 1 if registered == 0 & enrollhatr2 == 0

        //vselect's choice for AIC & AICC
        gen enrollhataic = 0
        replace enrollhataic = 1 if pr_enroll_aic > 0.4
        gen falseposaic = 0
        replace falseposaic = 1 if registered == 0 & enrollhataic == 1
        gen falsenegaic = 0
        replace falsenegaic = 1 if registered == 1 & enrollhataic == 0
        gen trueposaic = 0
        replace trueposaic = 1 if registered == 1 & enrollhataic == 1
        gen truenegaic = 0
        replace truenegaic = 1 if registered == 0 & enrollhataic == 0

        //vselect's choice for BIC
        gen enrollhatbic = 0
        replace enrollhatbic = 1 if pr_enroll_bic > 0.4
        gen falseposbic = 0
        replace falseposbic = 1 if registered == 0 & enrollhatbic == 1
        gen falsenegbic = 0
        replace falsenegbic = 1 if registered == 1 & enrollhatbic == 0
        gen trueposbic = 0
        replace trueposbic = 1 if registered == 1 & enrollhatbic == 1
        gen truenegbic = 0
        replace truenegbic = 1 if registered == 0 & enrollhatbic == 0

        //using all the variables
        gen enrollhatall = 0
```

```
replace enrollhatall = 1 if pr_enroll_all > 0.4
gen falseposall = 0
replace falseposall = 1 if registered == 0 & enrollhatall == 1
gen falsenegall = 0
replace falsenegall = 1 if registered == 1 & enrollhatall == 0
gen trueposall = 0
replace trueposall = 1 if registered == 1 & enrollhatall == 1
gen truenegall = 0
replace truenegall = 1 if registered == 0 & enrollhatall == 0

//Generating number of students that registered
gen numreg = registered == 1
replace numreg = sum(numreg)
replace numreg = numreg[_N]

//Generating number of students that didn't register
gen numnoreg = registered == 0
replace numnoreg = sum(numnoreg)
replace numnoreg = numnoreg[_N]

//Generating numbers for R^2
gen numtpr2 = trueposr2 == 1
replace numtpr2 = sum(numtpr2)
replace numtpr2 = numtpr2[_N]
gen numtnr2 = truenegr2 == 1
replace numtnr2 = sum(numtnr2)
replace numtnr2 = numtnr2[_N]
gen numfpr2 = falseposr2 == 1
replace numfpr2 = sum(numfpr2)
replace numfpr2 = numfpr2[_N]
gen numfnr2 = falsenegr2 == 1
replace numfnr2 = sum(numfnr2)
replace numfnr2 = numfnr2[_N]

//Generating numbers for AIC
gen numtpaic = trueposaic == 1
replace numtpaic = sum(numtpaic)
replace numtpaic = numtpaic[_N]
gen numtnaic = truenegaic == 1
replace numtnaic = sum(numtnaic)
replace numtnaic = numtnaic[_N]
gen numfpaic = falseposaic == 1
replace numfpaic = sum(numfpaic)
replace numfpaic = numfpaic[_N]
gen numfnaic = falsenegaic == 1
replace numfnaic = sum(numfnaic)
replace numfnaic = numfnaic[_N]

//Generating numbers for bic
gen numtpbic = trueposbic == 1
```

```
replace numtpbic = sum(numtpbic)
replace numtpbic = numtpbic[_N]
gen numtnbic = truenegbic == 1
replace numtnbic = sum(numtnbic)
replace numtnbic = numtnbic[_N]
gen numfpbic = falseposbic == 1
replace numfpbic = sum(numfpbic)
replace numfpbic = numfpbic[_N]
gen numfnbic = falsenegbic == 1
replace numfnbic = sum(numfnbic)
replace numfnbic = numfnbic[_N]

//Generating numbers using all variables
gen numtpall = trueposall == 1
replace numtpall = sum(numtpall)
replace numtpall = numtpall[_N]
gen numtnall = truenegall == 1
replace numtnall = sum(numtnall)
replace numtnall = numtnall[_N]
gen numfpall = falseposall == 1
replace numfpall = sum(numfpall)
replace numfpall = numfpall[_N]
gen numfnall = falsenegall == 1
replace numfnall = sum(numfnall)
replace numfnall = numfnall[_N]

//Creating percentages to be show true positive, true negative, etc.
gen percenttpr2 = numtpr2 / numreg
gen percenttnr2 = numtnr2 / numnoreg
gen percentfpr2 = numfpr2 / numnoreg
gen percentfnr2 = numfnr2 / numreg
gen percenttpaic = numtpaic / numreg
gen percenttnaic = numtnaic / numnoreg
gen percentfpaic = numfpaic / numnoreg
gen percentfnaic = numfnaic / numreg
gen percenttpbic = numtpbic / numreg
gen percenttnbic = numtnbic / numnoreg
gen percentfpbic = numfpbic / numnoreg
gen percentfnbic = numfnbic / numreg
gen percenttpall = numtpall / numreg
gen percenttnall = numtnall / numnoreg
gen percentfpall = numfpall / numnoreg
gen percentfnall = numfnall / numreg

//Displaying Results
dis percenttpr2
dis percenttnr2
dis percentfpr2
dis percentfnr2
dis percenttpaic
```

```
	dis percenttnaic
	dis percentfpaic
	dis percentfnaic
	dis percenttpbic
	dis percenttnbic
	dis percentfpbic
	dis percentfnbic
	dis percenttpall
	dis percenttnall
	dis percentfpall
	dis percentfnall

end //-------------predictpropensity----------------

/*******************************/

program vselectgpa

	clear

	use masterlist.dta

	//Finding the best fit
	vselect gpa TotalLen GrammerWrong WordLength NumUnique lexrich
syllables sentences fleschreading fleschgrade gpa2 gpa3 gpa4 gpa5 gpa6 gpa7 gpa8
gpa9 gpa10 gpa11 gpa12 gpa13 gpa14 gpa15 gpa16 gpa17 gpa18 gpa19 gpa20, best

end //--------------vselectgpa-------------------------------

/*******************************/

program text2gpa

	clear

	use masterlist.dta

	//Regressions based on GPA, needs to be changed for robustness
	//Also using random number assignment for out of sample predictions, can be
changed for robustness
	quiet regress gpa distance gpa14 gpa8 WordLength TotalLen gpa3
high_school_percentile gpa11 satmodel gpa6 gpa15 if random > .5
	quiet predict pr_abil_r2
	quiet regress gpa distance fleschreading WordLength TotalLen gpa3
high_school_percentile gpa11 satmodel gpa13 gpa15 if random > .5
	quiet predict pr_abil_aic
	quiet regress gpa syllables TotalLen high_school_percentile satmodel if
random > .5
	quiet predict pr_abil_bic
	quiet regress gpa TotalLen GrammerWrong WordLength NumUnique lexrich
```

```
syllables sentences fleschreading fleschgrade if random > .5
        quiet predict pr_abil_all
        quiet regress gpa TotalLen GrammerWrong WordLength NumUnique lexrich
sentences fleschreading fleschgrade if random > .5
        quiet predict pr_abil_working

        //Droping in sample
        drop if random > .5

        //Generating variable to show percentage I am off by
        gen off_r2 = ( pr_abil_r2 - gpa ) / gpa
        gen off_aic = ( pr_abil_aic - gpa ) / gpa
        gen off_bic = ( pr_abil_bic - gpa ) / gpa
        gen off_all = ( pr_abil_all - gpa ) / gpa
        gen off_working = ( pr_abil_working - gpa ) / gpa

        //Displaying Results
        sum off_r2
        sum off_aic
        sum off_bic
        sum off_all
        sum off_working

        //Mean Square Prediction Error Calculation
        gen msper2_1 = ( pr_abil_r2 - gpa )^2
        gen mspeaic_1= ( pr_abil_aic - gpa )^2
        gen mspebic_1 =( pr_abil_bic - gpa )^2
        gen mspeall_1 =( pr_abil_all - gpa )^2
        gen mspeworking_1 = ( pr_abil_working - gpa )^2

        egen msper2_2 = mean(msper2_1)
        egen mspeaic_2 = mean(mspeaic_1)
        egen mspebic_2 = mean(mspebic_1)
        egen mspeall_2 = mean(mspeall_1)
        egen mspeworking_2 = mean(mspeworking_1)

        gen msper2 = msper2_2 ^ 0.5
        gen mspeaic = mspeaic_2 ^ 0.5
        gen mspebic = mspebic_2 ^ 0.5
        gen mspeall = mspeall_2 ^ 0.5
        gen mspeworking = mspeworking_2 ^ 0.5

        sum msper2
        sum mspeaic
        sum mspebic
        sum mspeall
        sum mspeworking
```

```
end //----------------text2gpa-----------------------------------------

/*******************************/

program vselectsat

        clear

        use masterlist.dta

        //Finding the best fit
        vselect satmodel TotalLen GrammerWrong WordLength NumUnique lexrich
syllables sentences fleschreading fleschgrade, best

end //-----------------vselectsat----------------------------------------

        /*******************************/

program text2sat

        clear

        use masterlist.dta

        //Regressions based on GPA, needs to be changed for robustness
        //Also using random number assignment for out of sample predictions, can be
changed for robustness
        quiet regress satmodel NumUnique sentences WordLength TotalLen
fleschreading if random > .5
        quiet predict pr_abil_r2
        quiet regress satmodel NumUnique sentences WordLength TotalLen if
random > .5
        quiet predict pr_abil_aic
        quiet regress satmodel NumUnique sentences WordLength if random > .5
        quiet predict pr_abil_bic
        quiet regress gpa TotalLen GrammerWrong WordLength NumUnique lexrich
syllables sentences fleschreading fleschgrade if random > .5
        quiet predict pr_abil_all

        drop if random > .5

        //Generating variable to show percentage I am off by
        gen off_r2 = ( pr_abil_r2 - satmodel ) / satmodel
        gen off_aic = ( pr_abil_aic - satmodel ) / satmodel
        gen off_bic = ( pr_abil_bic - satmodel ) / satmodel
        gen off_all = ( pr_abil_all - satmodel ) / satmodel

        //Displaying Results
        sum off_r2
        sum off_aic
```

```
        sum off_bic
        sum off_all

        //Mean Square Prediction Error Calculation
        gen msper2_1 = ( pr_abil_r2 - gpa )^2
        gen mspeaic_1= ( pr_abil_aic - gpa )^2
        gen mspebic_1 =( pr_abil_bic - gpa )^2
        gen mspeall_1 =( pr_abil_all - gpa )^2
        gen mspeworking_1 = ( pr_abil_working - gpa )^2

        egen msper2_2 = mean(msper2_1)
        egen mspeaic_2 = mean(mspeaic_1)
        egen mspebic_2 = mean(mspebic_1)
        egen mspeall_2 = mean(mspeall_1)
        egen mspeworking_2 = mean(mspeworking_1)

        gen msper2 = msper2_2 ^ 0.5
        gen mspeaic = mspeaic_2 ^ 0.5
        gen mspebic = mspebic_2 ^ 0.5
        gen mspeall = mspeall_2 ^ 0.5
        gen mspeworking = mspeworking_2 ^ 0.5

        sum msper2
        sum mspeaic
        sum mspebic
        sum mspeall
        sum mspeworking

end //----------------------text2sat---------------------------------

/*******************************/

program sat2gpa

        clear

        use masterlist.dta

        quiet regress gpa satmodel if random > .5
        quiet predict gpahat

        drop if random > .5
        //drop if sat > 1200

        gen off = (gpahat - gpa)/gpa
        sum off

        //Mean Square Prediction Error Calculation
        gen msper2_1 = ( gpahat - gpa )^2
```

```
	egen msper2_2 = mean(msper2_1)

	gen msper2 = msper2_2 ^ 0.5

	sum msper2


end //----------------------sat2gpa----------------------------------

/*****************************/

program text2gpamarginal

	clear

	use masterlist.dta

	quiet regress gpa fleschreading fleschgrade sentences NumUnique syllables
lexrich TotalLen if random > .5
	quiet predict pr_gpa_r2
	quiet regress gpa fleschreading fleschgrade sentences NumUnique lexrich if
random > .5
	quiet predict pr_gpa_aic
	quiet regress gpa fleschreading TotalLen if random > .5
	quiet predict pr_gpa_bic
	quiet regress gpa TotalLen GrammerWrong WordLength NumUnique lexrich
syllables sentences fleschreading fleschgrade if random > .5
	quiet predict pr_gpa_all
	quiet regress satmodel NumUnique sentences WordLength TotalLen
fleschreading if random > .5
	quiet predict pr_sat_r2
	quiet regress satmodel NumUnique sentences WordLength TotalLen if
random > .5
	quiet predict pr_sat_aic
	quiet regress satmodel NumUnique sentences WordLength if random > .5
	quiet predict pr_sat_bic
	quiet regress gpa TotalLen GrammerWrong WordLength NumUnique lexrich
syllables sentences fleschreading fleschgrade if random > .5
	quiet predict pr_sat_all

	drop if random > .5
	//Using 50 points over cutoff
	drop if satmodel > 1250
	//Cutoff if 50% on high school rank, making marginal student 60%
	//drop if high_school_percentile > 60


	gen off_r2gpa = ( pr_gpa_r2 - gpa ) / gpa
	gen off_aicgpa = ( pr_gpa_aic - gpa ) / gpa
	gen off_bicgpa = ( pr_gpa_bic - gpa ) / gpa
```

```
        gen off_allgpa = ( pr_gpa_all - gpa ) / gpa
        gen off_r2sat = ( pr_sat_r2 - satmodel ) / satmodel
        gen off_aicsat = ( pr_sat_aic - satmodel ) / satmodel
        gen off_bicsat = ( pr_sat_bic - satmodel ) / satmodel
        gen off_allsat = ( pr_sat_all - satmodel ) / satmodel

        sum off_r2gpa
        sum off_aicgpa
        sum off_bicgpa
        sum off_allgpa
        sum off_r2sat
        sum off_aicsat
        sum off_bicsat
        sum off_allsat

end //-----------------------text2gpamarginal------------------------------

program openanswerconvert

        insheet using additionaloutput.csv
        save additionalstatement.dta

        rename id random_ID
        rename additional_info_statement addinfo
        rename v3 addinfolen
        rename v5 addGrammerWrong
        rename v6 addWordLength
        rename v7 addNumUnique
        rename v8 addlexrich
        rename v9 addcommitmentcounter
        rename v10 addcommitmentpercent
        rename v11 addfaithcounter
        rename v12 addfaithpercent
        rename v13 addbaylorcounter
        rename v14 addbaylorpercent
        rename v15 addtotalbible
        rename v16 addbiblepercent
        rename v17 addadmincounter
        rename v18 addadminpercent
        rename v19 addsyllables
        rename v20 addsentences
        rename v21 addfleschreading
        rename v22 addfleschgrade
        drop v4

        save additionalstatement.dta, replace

        clear
        use masterlist.dta
```

```
        merge m:1 random_ID using additionalstatement.dta, nogen

        save masterwithopen.dta

end //-------------------------openanswerconvert---------------------------

/*******************************/

program textsat2gpa

        clear

        use masterlist.dta

        //Regressions based on GPA, needs to be changed for robustness
        //Also using random number assignment for out of sample predictions, can be
changed for robustness
        quiet regress gpa satmodel fleschreading fleschgrade sentences NumUnique
syllables lexrich TotalLen if random > .5
        quiet predict pr_abil_r2
        quiet regress gpa satmodel fleschreading fleschgrade sentences NumUnique
lexrich if random > .5
        quiet predict pr_abil_aic
        quiet regress gpa satmodel fleschreading TotalLen if random > .5
        quiet predict pr_abil_bic
        quiet regress gpa satmodel TotalLen GrammerWrong WordLength
NumUnique lexrich syllables sentences fleschreading fleschgrade if random > .5
        quiet predict pr_abil_all
        quiet regress gpa satmodel fleschreading fleschgrade syllables TotalLen
sentences lexrich NumUnique if random > .5
        quiet predict pr_abil_working

        //Droping in sample
        drop if random > .5

        //Generating variable to show percentage I am off by
        gen off_r2 = ( pr_abil_r2 - gpa ) / gpa
        gen off_aic = ( pr_abil_aic - gpa ) / gpa
        gen off_bic = ( pr_abil_bic - gpa ) / gpa
        gen off_all = ( pr_abil_all - gpa ) / gpa
        gen off_working = ( pr_abil_working - gpa ) / gpa

        //Displaying Results
        sum off_r2
        sum off_aic
        sum off_bic
        sum off_all
        sum off_working

end //----------------textsat2gpa-------------------------------------------
```

```
/*******************************/

program hsp2gpa

	clear

	use masterlist.dta

	quiet regress gpa high_school_percentile if random > .5
	quiet predict gpahat

	drop if random > .5

	gen off = (gpahat - gpa)/gpa
	sum off

end //----------------------hsp2gpa-----------------------------------

/*******************************/

program textsathsp2gpa

	clear

	use masterlist.dta

	//Regressions based on GPA, needs to be changed for robustness
	//Also using random number assignment for out of sample predictions, can be
changed for robustness
	quiet regress gpa satmodel high_school_percentile fleschreading fleschgrade
sentences NumUnique syllables lexrich TotalLen if random > .5
	quiet predict pr_abil_r2
	quiet regress gpa satmodel high_school_percentile fleschreading fleschgrade
sentences NumUnique lexrich if random > .5
	quiet predict pr_abil_aic
	quiet regress gpa satmodel high_school_percentile fleschreading TotalLen if
random > .5
	quiet predict pr_abil_bic
	quiet regress gpa satmodel high_school_percentile TotalLen GrammerWrong
WordLength NumUnique lexrich syllables sentences fleschreading fleschgrade if
random > .5
	quiet predict pr_abil_all
	quiet regress gpa high_school_percentile satmodel NumUnique lexrich
fleschreading syllables GrammerWrong sentences WordLength fleschgrade if random
> .5
	quiet predict pr_abil_working

	//Droping in sample
```

```
drop if random > .5
*Add below code in for marginal work
//drop if satmodel > 1200

//Generating variable to show percentage I am off by
gen off_r2 = ( pr_abil_r2 - gpa ) / gpa
gen off_aic = ( pr_abil_aic - gpa ) / gpa
gen off_bic = ( pr_abil_bic - gpa ) / gpa
gen off_all = ( pr_abil_all - gpa ) / gpa
gen off_working = ( pr_abil_working - gpa ) / gpa

//Displaying Results
sum off_r2
sum off_aic
sum off_bic
sum off_all
sum off_working

//Mean Square Prediction Error Calculation
gen msper2_1 = ( pr_abil_r2 - gpa )^2
gen mspeaic_1= ( pr_abil_aic - gpa )^2
gen mspebic_1 =( pr_abil_bic - gpa )^2
gen mspeall_1 =( pr_abil_all - gpa )^2
gen mspeworking_1 = ( pr_abil_working - gpa )^2

egen msper2_2 = mean(msper2_1)
egen mspeaic_2 = mean(mspeaic_1)
egen mspebic_2 = mean(mspebic_1)
egen mspeall_2 = mean(mspeall_1)
egen mspeworking_2 = mean(mspeworking_1)

gen msper2 = msper2_2 ^ 0.5
gen mspeaic = mspeaic_2 ^ 0.5
gen mspebic = mspebic_2 ^ 0.5
gen mspeall = mspeall_2 ^ 0.5
gen mspeworking = mspeworking_2 ^ 0.5

sum msper2
sum mspeaic
sum mspebic
sum mspeall
sum mspeworking

end //----------------textsathsp2gpa-------------------------------------------

/*******************************/

program vselectretention

        clear
```

```
		use masterlist.dta

		quiet regress gpa fleschreading fleschgrade sentences NumUnique
syllables lexrich TotalLen
		quiet predict pr_abil_r2

		//Those that were not admitted are not being used
		drop if drop_semester == .
		drop if drop_year == .

		//The vselect will be fixed on a single year, this will need to be
changed for robustness checks
		vselect drop_year pr_abil_r2 orientation_sched_ind budget_amt
gross_need academic_index high_school_percentile total_contacts self_init_cntcts
solicited_cntcts referral_cntcts campus_visit BUhonors satmodel dep_span legacy
distance visited hsrate mrate initrate religionrate tot_loan tot_schol tot_work tot_grant
efc merit_schol need_schol commitmentpercent commitmentcounter faithcounter
faithpercent baylorcounter baylorpercent totalbible biblepercent admincounter if
newrandom > .5, best


end //-------------veselectretention----------------

/*******************************/

program predictretention

	clear

	use masterlist.dta

	//Logits run based on the vselect recommendations.  These will need to be
changed for robustness check
	quiet logit drop_semester orientation_sched_ind dep_span
high_school_percentile academic_index satmodel need_schol campus_visit
biblepercent BUhonors baylorpercent tot_loan tot_schol hsrate merit_schol tot_work
commitmentcounter faithpercent if newrandom > .5, difficult technique(nr bhhh dfp
bfgs)
	quiet predict pr_semester_r2, pr
	quiet logit drop_semester orientation_sched_ind dep_span
high_school_percentile campus_visit biblepercent BUhonors baylorpercent tot_loan
hsrate totalbible faithcounter if newrandom > .5, difficult technique(nr bhhh dfp bfgs)
	quiet predict pr_semester_aic, pr
	quiet logit drop_semester dep_span if newrandom > .5, difficult technique(nr
bhhh dfp bfgs)
	quiet predict pr_semester_bic, pr
	quiet logit drop_semester orientation_sched_ind dep_span
high_school_percentile campus_visit biblepercent baylorpercent tot_loan if
newrandom > .5, difficult technique(nr bhhh dfp bfgs)
```

```
quiet predict pr_semester_aicc, pr


quiet logit drop_year BUhonors budget_amt distance tot_schol need_schol
mrate merit_schol commitmentpercent high_school_percentile tot_loan
self_init_cntcts initrate orientation_sched_ind gross_need if newrandom > .5, difficult
technique(nr bhhh dfp bfgs)
quiet predict pr_year_r2, pr
quiet logit drop_year BUhonors budget_amt distance mrate
commitmentpercent tot_loan academic_index self_init_cntcts initrate tot_work if
newrandom > .5, difficult technique(nr bhhh dfp bfgs)
quiet predict pr_year_aic, pr
quiet logit drop_year self_init_cntcts gross_need if newrandom > .5, difficult
technique(nr bhhh dfp bfgs)
quiet predict pr_year_bic, pr

//Droping term used in regressions to get out of sample predictions
drop if newrandom > .5
drop if drop_semester == .

//vselect's choice of R^2
gen semesterhatr2 = 0
replace semesterhatr2 = 1 if pr_semester_r2 > 0.3
gen falseposr2 = 0
replace falseposr2 = 1 if drop_semester == 0 & semesterhatr2 == 1
gen falsenegr2 = 0
replace falsenegr2 = 1 if drop_semester == 1 & semesterhatr2 == 0
gen trueposr2 = 0
replace trueposr2 = 1 if drop_semester == 1 & semesterhatr2 == 1
gen truenegr2 = 0
replace truenegr2 = 1 if drop_semester == 0 & semesterhatr2 == 0

//vselect's choice for AIC
gen semesterhataic = 0
replace semesterhataic = 1 if pr_semester_aic > 0.3
gen falseposaic = 0
replace falseposaic = 1 if drop_semester == 0 & semesterhataic == 1
gen falsenegaic = 0
replace falsenegaic = 1 if drop_semester == 1 & semesterhataic == 0
gen trueposaic = 0
replace trueposaic = 1 if drop_semester == 1 & semesterhataic == 1
gen truenegaic = 0
replace truenegaic = 1 if drop_semester == 0 & semesterhataic == 0

//vselect's choice for BIC
gen semesterhatbic = 0
replace semesterhatbic = 1 if pr_semester_bic > 0.3
gen falseposbic = 0
replace falseposbic = 1 if drop_semester == 0 & semesterhatbic == 1
gen falsenegbic = 0
```

```
replace falsenegbic = 1 if drop_semester == 1 & semesterhatbic == 0
gen trueposbic = 0
replace trueposbic = 1 if drop_semester == 1 & semesterhatbic == 1
gen truenegbic = 0
replace truenegbic = 1 if drop_semester == 0 & semesterhatbic == 0

//vselect's choice for AICC
gen semesterhataicc = 0
replace semesterhataicc = 1 if pr_semester_aicc > 0.3
gen falseposall = 0
replace falseposall = 1 if registered == 0 & semesterhataicc == 1
gen falsenegall = 0
replace falsenegall = 1 if registered == 1 & semesterhataicc == 0
gen trueposall = 0
replace trueposall = 1 if registered == 1 & semesterhataicc == 1
gen truenegall = 0
replace truenegall = 1 if registered == 0 & semesterhataicc == 0


//Generating number of students that registered
gen numsemester = drop_semester == 1
replace numsemester = sum(numsemester)
replace numsemester = numsemester[_N]


//Generating number of students that didn't register
gen numnosemester = drop_semester == 0
replace numnosemester = sum(numnosemester)
replace numnosemester = numnosemester[_N]

//Generating numbers for R^2
gen numtpr2 = trueposr2 == 1
replace numtpr2 = sum(numtpr2)
replace numtpr2 = numtpr2[_N]
gen numtnr2 = truenegr2 == 1
replace numtnr2 = sum(numtnr2)
replace numtnr2 = numtnr2[_N]
gen numfpr2 = falseposr2 == 1
replace numfpr2 = sum(numfpr2)
replace numfpr2 = numfpr2[_N]
gen numfnr2 = falsenegr2 == 1
replace numfnr2 = sum(numfnr2)
replace numfnr2 = numfnr2[_N]

//Generating numbers for AIC
gen numtpaic = trueposaic == 1
replace numtpaic = sum(numtpaic)
replace numtpaic = numtpaic[_N]
gen numtnaic = truenegaic == 1
replace numtnaic = sum(numtnaic)
```

```
replace numtnaic = numtnaic[_N]
gen numfpaic = falseposaic == 1
replace numfpaic = sum(numfpaic)
replace numfpaic = numfpaic[_N]
gen numfnaic = falsenegaic == 1
replace numfnaic = sum(numfnaic)
replace numfnaic = numfnaic[_N]

//Generating numbers for bic
gen numtpbic = trueposbic == 1
replace numtpbic = sum(numtpbic)
replace numtpbic = numtpbic[_N]
gen numtnbic = truenegbic == 1
replace numtnbic = sum(numtnbic)
replace numtnbic = numtnbic[_N]
gen numfpbic = falseposbic == 1
replace numfpbic = sum(numfpbic)
replace numfpbic = numfpbic[_N]
gen numfnbic = falsenegbic == 1
replace numfnbic = sum(numfnbic)
replace numfnbic = numfnbic[_N]

//Generating numbers using all variables
gen numtpall = trueposall == 1
replace numtpall = sum(numtpall)
replace numtpall = numtpall[_N]
gen numtnall = truenegall == 1
replace numtnall = sum(numtnall)
replace numtnall = numtnall[_N]
gen numfpall = falseposall == 1
replace numfpall = sum(numfpall)
replace numfpall = numfpall[_N]
gen numfnall = falsenegall == 1
replace numfnall = sum(numfnall)
replace numfnall = numfnall[_N]

//Creating percentages to be show true positive, true negative, etc.
gen percenttpr2 = numtpr2 / numsemester
gen percenttnr2 = numtnr2 / numnosemester
gen percentfpr2 = numfpr2 / numnosemester
gen percentfnr2 = numfnr2 / numsemester
gen percenttpaic = numtpaic / numsemester
gen percenttnaic = numtnaic / numnosemester
gen percentfpaic = numfpaic / numnosemester
gen percentfnaic = numfnaic / numsemester
gen percenttpbic = numtpbic / numsemester
gen percenttnbic = numtnbic / numnosemester
gen percentfpbic = numfpbic / numnosemester
gen percentfnbic = numfnbic / numsemester
gen percenttpall = numtpall / numsemester
```

```
gen percenttnall = numtnall / numnosemester
gen percentfpall = numfpall / numnosemester
gen percentfnall = numfnall / numsemester

//Displaying Results
dis percenttpr2
dis percenttnr2
dis percentfpr2
dis percentfnr2
dis percenttpaic
dis percenttnaic
dis percentfpaic
dis percentfnaic
dis percenttpbic
dis percenttnbic
dis percentfpbic
dis percentfnbic
dis percenttpall
dis percenttnall
dis percentfpall
dis percentfnall


gen yearhatr2 = 0
replace yearhatr2 = 1 if pr_year_r2 > 0.3
gen falseposr22 = 0
replace falseposr22 = 1 if drop_year == 0 & yearhatr2 == 1
gen falsenegr22 = 0
replace falsenegr22 = 1 if drop_year == 1 & yearhatr2 == 0
gen trueposr22 = 0
replace trueposr22 = 1 if drop_year == 1 & yearhatr2 == 1
gen truenegr22 = 0
replace truenegr22 = 1 if drop_year == 0 & yearhatr2 == 0

gen yearhataic = 0
replace yearhataic = 1 if pr_year_aic > 0.3
gen falseposaic2 = 0
replace falseposaic2 = 1 if drop_year == 0 & yearhataic == 1
gen falsenegaic2 = 0
replace falsenegaic2 = 1 if drop_year == 1 & yearhataic == 0
gen trueposaic2 = 0
replace trueposaic2 = 1 if drop_year == 1 & yearhataic == 1
gen truenegaic2 = 0
replace truenegaic2 = 1 if drop_year == 0 & yearhataic == 0

gen yearhatbic = 0
replace yearhatbic = 1 if pr_year_bic > 0.3
gen falseposbic2 = 0
replace falseposbic2 = 1 if drop_year == 0 & yearhatbic == 1
gen falsenegbic2 = 0
```

```
replace falsenegbic2 = 1 if drop_year == 1 & yearhatbic == 0
gen trueposbic2 = 0
replace trueposbic2 = 1 if drop_year == 1 & yearhatbic == 1
gen truenegbic2 = 0
replace truenegbic2 = 1 if drop_year == 0 & yearhatbic == 0

gen numyear = drop_year == 1
replace numyear = sum(numyear)
replace numyear = numyear[_N]

gen numnoyear = drop_year == 0
replace numnoyear = sum(numnoyear)
replace numnoyear = numnoyear[_N]

gen numtpr22 = trueposr22 == 1
replace numtpr22 = sum(numtpr22)
replace numtpr22 = numtpr22[_N]
gen numtnr22 = truenegr22 == 1
replace numtnr22 = sum(numtnr22)
replace numtnr22 = numtnr22[_N]
gen numfpr22 = falseposr22 == 1
replace numfpr22 = sum(numfpr22)
replace numfpr22 = numfpr22[_N]
gen numfnr22 = falsenegr22 == 1
replace numfnr22 = sum(numfnr22)
replace numfnr22 = numfnr22[_N]

gen numtpaic2 = trueposaic2 == 1
replace numtpaic2 = sum(numtpaic2)
replace numtpaic2 = numtpaic2[_N]
gen numtnaic2 = truenegaic2 == 1
replace numtnaic2 = sum(numtnaic2)
replace numtnaic2 = numtnaic2[_N]
gen numfpaic2 = falseposaic2 == 1
replace numfpaic2 = sum(numfpaic2)
replace numfpaic2 = numfpaic2[_N]
gen numfnaic2 = falsenegaic2 == 1
replace numfnaic2 = sum(numfnaic2)
replace numfnaic2 = numfnaic2[_N]

gen numtpbic2 = trueposbic2 == 1
replace numtpbic2 = sum(numtpbic2)
replace numtpbic2 = numtpbic2[_N]
gen numtnbic2 = truenegbic2 == 1
replace numtnbic2 = sum(numtnbic2)
replace numtnbic2 = numtnbic2[_N]
gen numfpbic2 = falseposbic2 == 1
replace numfpbic2 = sum(numfpbic2)
replace numfpbic2 = numfpbic2[_N]
gen numfnbic2 = falsenegbic2 == 1
```

```
        replace numfnbic2 = sum(numfnbic2)
        replace numfnbic2 = numfnbic2[_N]

        gen percenttpr22 = numtpr22 / numyear
        gen percenttnr22 = numtnr22 / numnoyear
        gen percentfpr22 = numfpr22 / numnoyear
        gen percentfnr22 = numfnr22 / numyear
        gen percenttpaic2 = numtpaic2 / numyear
        gen percenttnaic2 = numtnaic2 / numnoyear
        gen percentfpaic2 = numfpaic2 / numnoyear
        gen percentfnaic2 = numfnaic2 / numyear
        gen percenttpbic2 = numtpbic2 / numyear
        gen percenttnbic2 = numtnbic2 / numnoyear
        gen percentfpbic2 = numfpbic2 / numnoyear
        gen percentfnbic2 = numfnbic2 / numyear

        dis percenttpr22
        dis percenttnr22
        dis percentfpr22
        dis percentfnr22
        dis percenttpaic2
        dis percenttnaic2
        dis percentfpaic2
        dis percentfnaic2
        dis percenttpbic2
        dis percenttnbic2
        dis percentfpbic2
        dis percentfnbic2

end //-------------predictretention----------------

program subgroupingsat

        clear

        use masterlist.dta

        quiet regress gpa distance fleschreading WordLength TotalLen gpa3
high_school_percentile gpa11 satmodel gpa13 gpa15 if random > .3
        quiet predict pr_abil_aic
        quiet regress gpa satmodel if random > .3
        quiet predict pr_gpa_sat
        quiet regress gpa newrandom if random > .3
        quiet predict pr_gpa_random

        //Droping in sample
        drop if random > .3
        drop if gpa == .

        //Generating MSPE
```

```
gen mspeaic_1= ( pr_abil_aic - gpa )^2
egen mspeaic_2 = mean(mspeaic_1)
gen mspeaic = mspeaic_2 ^ 0.5

gen mspesat_1= ( pr_gpa_sat - gpa )^2
egen mspesat_2 = mean(mspesat_1)
gen mspesat = mspesat_2 ^ 0.5

gen msperan_1= ( pr_gpa_random - gpa )^2
egen msperan_2 = mean(msperan_1)
gen msperan = msperan_2 ^ 0.5

//Creating SAT Trenches
gen rank = 1 if satmodel > 1000 & satmodel < 1100
replace rank = 2 if satmodel >= 1100 & satmodel < 1200
replace rank = 3 if satmodel >= 1200 & satmodel < 1300
replace rank = 4 if satmodel >= 1300 & satmodel < 1400
replace rank = 5 if satmodel >= 1400 & satmodel < 1500
replace rank = 6 if satmodel >= 1500

//Mean Square Prediction Error Calculation
gen errors = pr_abil_aic - gpa
gen errors_sq = errors ^ 2

egen errors_avg_1 = mean(errors_sq) if rank == 1
egen errors_avg_2 = mean(errors_sq) if rank == 2
egen errors_avg_3 = mean(errors_sq) if rank == 3
egen errors_avg_4 = mean(errors_sq) if rank == 4
egen errors_avg_5 = mean(errors_sq) if rank == 5
egen errors_avg_6 = mean(errors_sq) if rank == 6

gen rmspe = errors_avg_1 ^ .5 if rank == 1
replace rmspe = errors_avg_2 ^ .5 if rank == 2
replace rmspe = errors_avg_3 ^ .5 if rank == 3
replace rmspe = errors_avg_4 ^ .5 if rank == 4
replace rmspe = errors_avg_5 ^ .5 if rank == 5
replace rmspe = errors_avg_6 ^ .5 if rank == 6

sum pr_abil_aic pr_gpa_sat gpa
sum pr_abil_aic pr_gpa_sat gpa if rank == 1
sum pr_abil_aic pr_gpa_sat gpa if rank == 2
sum pr_abil_aic pr_gpa_sat gpa if rank == 3
sum pr_abil_aic pr_gpa_sat gpa if rank == 4
sum pr_abil_aic pr_gpa_sat gpa if rank == 5
sum pr_abil_aic pr_gpa_sat gpa if rank == 6

sum rmspe mspesat msperan

end // subgroupingsat
```

```
program regressiononlevels

	clear

	use masterlist.dta

	//Creating SAT Trenches
	gen rank = 1 if satmodel < 1050
	replace rank = 2 if satmodel >= 1050 & satmodel < 1150
	replace rank = 3 if satmodel >= 1150 & satmodel < 1250
	replace rank = 4 if satmodel >= 1250 & satmodel < 1350
	replace rank = 5 if satmodel >= 1350 & satmodel < 1450
	replace rank = 6 if satmodel >= 1450

	regress gpa syllables gpa15 gpa12 gpa7 gpa2 GrammerWrong gpa16
sentences gpa19 gpa10 gpa14 gpa17 if rank == 1 & random > .5
	quiet predict pr_abil_aic1
	regress gpa syllables gpa15 gpa12 gpa7 gpa2 GrammerWrong gpa16
sentences gpa19 gpa10 gpa14 gpa17 if rank == 2 & random > .5
	quiet predict pr_abil_aic2
	regress gpa syllables gpa15 gpa12 gpa7 gpa2 GrammerWrong gpa16
sentences gpa19 gpa10 gpa14 gpa17 if rank == 3 & random > .5
	quiet predict pr_abil_aic3
	regress gpa syllables gpa15 gpa12 gpa7 gpa2 GrammerWrong gpa16
sentences gpa19 gpa10 gpa14 gpa17 if rank == 4 & random > .5
	quiet predict pr_abil_aic4
	regress gpa syllables gpa15 gpa12 gpa7 gpa2 GrammerWrong gpa16
sentences gpa19 gpa10 gpa14 gpa17 if rank == 5 & random > .5
	quiet predict pr_abil_aic5
	regress gpa syllables gpa15 gpa12 gpa7 gpa2 GrammerWrong gpa16
sentences gpa19 gpa10 gpa14 gpa17 if rank == 6 & random > .5
	quiet predict pr_abil_aic6
	quiet regress gpa syllables gpa15 gpa12 gpa7 gpa2 GrammerWrong gpa16
sentences gpa19 gpa10 gpa14 gpa17 if random > .5
	quiet predict pr_abil_aic

	quiet regress gpa satmodel if rank == 1 & random > .5
	quiet predict pr_gpa_sat1
	quiet regress gpa satmodel if rank == 2 & random > .5
	quiet predict pr_gpa_sat2
	quiet regress gpa satmodel if rank == 3 & random > .5
	quiet predict pr_gpa_sat3
	quiet regress gpa satmodel if rank == 4 & random > .5
	quiet predict pr_gpa_sat4
	quiet regress gpa satmodel if rank == 5 & random > .5
	quiet predict pr_gpa_sat5
	quiet regress gpa satmodel if rank == 6 & random > .5
	quiet predict pr_gpa_sat6
	quiet regress gpa satmodel if random > .5
	quiet predict pr_gpa_sat
```

```
quiet regress gpa newrandom if random > .5
quiet predict pr_gpa_random

//Droping in sample
drop if gpa == .
drop if random > .5

//Mean Square Prediction Error Calculation
gen errors = pr_abil_aic - gpa
gen errors_sq = errors ^ 2

egen errors_avg_1 = mean(errors_sq) if rank == 1
egen errors_avg_2 = mean(errors_sq) if rank == 2
egen errors_avg_3 = mean(errors_sq) if rank == 3
egen errors_avg_4 = mean(errors_sq) if rank == 4
egen errors_avg_5 = mean(errors_sq) if rank == 5
egen errors_avg_6 = mean(errors_sq) if rank == 6

gen rmspe = errors_avg_1 ^ .5 if rank == 1
replace rmspe = errors_avg_2 ^ .5 if rank == 2
replace rmspe = errors_avg_3 ^ .5 if rank == 3
replace rmspe = errors_avg_4 ^ .5 if rank == 4
replace rmspe = errors_avg_5 ^ .5 if rank == 5
replace rmspe = errors_avg_6 ^ .5 if rank == 6

//Generating MSPE
gen mspeaic_1= ( pr_abil_aic - gpa )^2
egen mspeaic_2 = mean(mspeaic_1)
gen mspeaic = mspeaic_2 ^ 0.5

gen mspesat_1= ( pr_gpa_sat - gpa )^2
egen mspesat_2 = mean(mspesat_1)
gen mspesat = mspesat_2 ^ 0.5

gen msperan_1= ( pr_gpa_random - gpa )^2
egen msperan_2 = mean(msperan_1)
gen msperan = msperan_2 ^ 0.5

//
sum pr_abil_aic pr_gpa_sat gpa
sum pr_abil_aic pr_gpa_sat gpa if rank == 1
sum pr_abil_aic pr_gpa_sat gpa if rank == 2
sum pr_abil_aic pr_gpa_sat gpa if rank == 3
sum pr_abil_aic pr_gpa_sat gpa if rank == 4
sum pr_abil_aic pr_gpa_sat gpa if rank == 5
sum pr_abil_aic pr_gpa_sat gpa if rank == 6

sum pr_abil_aic1 pr_gpa_sat1 gpa if rank == 1
sum pr_abil_aic2 pr_gpa_sat2 gpa if rank == 2
```

```
        sum pr_abil_aic3 pr_gpa_sat3 gpa if rank == 3
        sum pr_abil_aic4 pr_gpa_sat4 gpa if rank == 4
        sum pr_abil_aic5 pr_gpa_sat5 gpa if rank == 5
        sum pr_abil_aic6 pr_gpa_sat6 gpa if rank == 6

        gen predicted_gpa = pr_abil_aic1 if rank == 1
        replace predicted_gpa = pr_abil_aic2 if rank == 2
        replace predicted_gpa = pr_abil_aic3 if rank == 3
        replace predicted_gpa = pr_abil_aic4 if rank == 4
        replace predicted_gpa = pr_abil_aic5 if rank == 5
        replace predicted_gpa = pr_abil_aic6 if rank == 6

        gen predicted_gpa_sat = pr_gpa_sat1 if rank == 1
        replace predicted_gpa_sat = pr_gpa_sat2 if rank == 2
        replace predicted_gpa_sat = pr_gpa_sat3 if rank == 3
        replace predicted_gpa_sat = pr_gpa_sat4 if rank == 4
        replace predicted_gpa_sat = pr_gpa_sat5 if rank == 5
        replace predicted_gpa_sat = pr_gpa_sat6 if rank == 6


        tabstat predicted_gpa predicted_gpa_sat gpa, by(rank)

        sum rmspe mspesat msperan

end // regressiononlevels

program bootstrapping

        clear

        use masterlist.dta

        regress gpa syllables gpa15 gpa12 gpa7 gpa2 GrammerWrong gpa16
sentences gpa19 gpa10 gpa14 gpa17

        set more off

        gen syllablescoef = _b[syllables]
        gen gpa15coef = _b[gpa15]
        gen gpa12coef = _b[gpa12]
        gen gpa7coef = _b[gpa7]
        gen gpa2coef = _b[gpa2]
        gen grammerwrongcoef = _b[GrammerWrong]
        gen gpa16coef = _b[gpa16]
        gen sentencescoef = _b[sentences]
        gen gpa19coef = _b[gpa19]
        gen gpa10coef = _b[gpa10]
        gen gpa14coef = _b[gpa14]
        gen gpa17coef = _b[gpa17]
        gen concoef = _b[_cons]
```

```
        gen syllablessd = _se[syllables]
        gen gpa15sd = _se[gpa15]
        gen gpa12sd = _se[gpa12]
        gen gpa7sd = _se[gpa7]
        gen gpa2sd = _se[gpa2]
        gen grammerwrongsd = _se[GrammerWrong]
        gen gpa16sd = _se[gpa16]
        gen sentencescsd = _se[sentences]
        gen gpa19sd = _se[gpa19]
        gen gpa10sd = _se[gpa10]
        gen gpa14sd = _se[gpa14]
        gen gpa17sd = _se[gpa17]
        gen consd = _se[_cons]


        gen obs = _n
        gen runningtotal = 0


forvalues i = 1/6962 {
        drawnorm syllran, means(-.0128083) sds(.0034625)
        drawnorm gpa15ran, means(.0211936) sds(.0050212)
        drawnorm gpa12ran, means(.3358554) sds(.1017359)
        drawnorm gpa7ran, means(.0002221) sds(.0001486)
        drawnorm gpa2ran, means(-.0000143) sds(4.30e-06)
        drawnorm gwrongran, means(-32.48605) sds(10.36602)
        drawnorm gpa16ran, means(9.28e-06) sds(3.15e-06)
        drawnorm sentran, means(-.0923634) sds(.0265255)
        drawnorm gpa19ran, means(-3.18e-06) sds(7.47e-07)
        drawnorm gpa10ran, means(-.0040628) sds(.0009783)
        drawnorm gpa14ran, means(.0209936) sds(.0057428)
        drawnorm gpa17ran, means(.0116991) sds(.0036935)
        drawnorm conran, means(6.176882) sds(.8541035)

        gen gpaexpect`i' = conran + syllran*syllables + gpa15ran*gpa15 +
gpa12ran*gpa12 + gpa7ran*gpa7 + gpa2ran*gpa2 + gwrongran*GrammerWrong +
gpa16ran*gpa16 + sentran*sentences + gpa19ran*gpa19 + gpa10ran*gpa10 +
gpa14ran*gpa14 + gpa17ran*gpa17

        drop syllran
        drop gpa15ran
        drop gpa12ran
        drop gpa7ran
        drop gpa2ran
        drop gwrongran
        drop gpa16ran
        drop sentran
        drop gpa19ran
        drop gpa10ran
```

```
        drop gpa14ran
        drop gpa17ran
        drop conran

}
set more off
forvalues i = 1/6962{
        replace runningtotal = runningtotal + gpaexpect`i'

}
set more off
gen meanstuff = runningtotal / 6962
gen makevar = 0
gen herevar = 0
forvalues i = 1/6962{
        replace herevar = (gpaexpect`i' - meanstuff)^2
        replace makevar = makevar + herevar
}
replace makevar = makevar / 6962

gen makesd = sqrt(makevar)
gen citop = meanstuff + (1.96 * (makesd / sqrt(6962)))
gen cibottom = meanstuff - (1.96 * (makesd / sqrt(6962)))
```

APPENDIX G

Variable Summary Statistics

Table 13: Summary Statistics of Variables

Variable	Obs	Mean	Std. Dev.	Min	Max
Essay1Len	20167	231.6866	122.4415	1	549
Essay2Len	20167	179.777	116.146	1	533
TotalLen	20167	411.4636	219.5576	2	1027
GrammerWrong	20167	0.02201	0.041947	0.005842	3
WordLength	20167	4.599318	17.57521	0.5	2500
NumUnique	20167	185.2585	81.96735	1	450
lexrich	20167	2.125308	0.350667	1	4.939227
commitment~r	20167	5.39401	4.112889	0	38
commitment~t	20167	0.013385	0.008699	0	0.078431
faithcounter	20167	8.520058	5.852545	0	49
faithpercent	20167	0.021392	0.011235	0	0.2
baylorcoun~r	20167	7.142956	4.522037	0	38
baylorperc~t	20167	0.019172	0.009997	0	0.166667
totalbible	20167	0.172063	0.497586	0	8
biblepercent	20167	0.000395	0.001262	0	0.027778
admincounter	20167	4.551594	3.505964	0	26
adminpercent	20167	0.011134	0.007311	0	0.078431
syllables	20167	554.5818	301.1856	1	1307
sentences	20167	20.54321	11.4979	1	96
fleschread~g	20167	72.06665	10.93531	3.344545	162.505
fleschgrade	20167	8.332435	2.324321	-8.91	32.25407

BIBLIOGRAPHY

Atkinson, Richard. "Achievement Versus Aptitude Tests in College Admissions." Issues in Science and Technology 18, no. 2 (2001) http://works.bepress.com/cgi/viewcontent.cgi?article=1027&context=richard_atkinson(accessed May 28, 2014).

Balfour, Stephen. "Assessing Writing in MOOCs: Automated Essay Scoring and Calibrated Peer Review." Research & Practice in Assessment 8 (2013): 40-48.

Bridgeman, Brent. "Essays and Multiple-Choice Tests as Predictors of College Freshman GPA." Research in Higher Education 32, no. 3 (1991): 319-332.

Dikli, Semire. "Automated Essay Scoring.ò Turkish Online Journal of Distance Education 7, no. 1 (2006): 49-62.

Ericsson, Patricia Freitag and Haswell, Richard H., "Machine Scoring of Student Essays: Truth and Consequences" (2006). All USU Press Publications. Book 139. http://digitalcomraons.usu.edu/usupress_ pubs/ 139 (accessed May 29, 2014).

Foltz, P.W., Laham, D. and Landauer, T.K. ñAutomated Essay Scoring: Applications to Educational Technology.ò In B. Collis & R. Oliver (Eds.), Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (1999*)*: 939-944.

Geiser, Saul, and Studley, Roger. "UC and the SAT: Predictive Validity and Differential Impact of the SAT I and SAT II at the University of California." Educational Assessment 8, no. 1 (2002): 1-26.

Jaschik, Scott. "The (Needless?) Frenzy." Inside Higher Ed. http://www. insidehighered. com/news/2012/11/29/study-documents-admissions-trends-over-last-10-years# sthash.xo5qUbQK.dpbs (accessed May 20, 2014).

Kellogg, Ronald and Bascom, Raulerson III. "Improving the Writing Skills of College Students." Psychonomic Bulletin & Review 14, no. 2 (2007): 237-242.

Page, Ellis. "The Use of the Computer in Analyzing Student Essays." International Review of Education 14, no. 2 (1968): 210-225.

Thomas, E., Ruznik, G. and Dawes, W. ñUsing Predictive Modeling to Target Student Recruitment: Theory and Practice.ò AIR Professional File (2001): 78.

Van Dyke, Sarah. Predicting the Probability of Enrollment of Accepted Applicants at Baylor University. (2014).