Predicting Students' First-year Success at UC Santa Cruz

Hairong Wu, Kevin Doyle, Connor McNeill

Preface

We use different tools to explore a problem described below. Those separate explorations developed separate questions, and ultimately different solutions. The first approach begins with a proper introduction to the problem. The second approach gets straight to the methods.

First Approach:

Abstract: In order to accurately estimate student's' GPA on testing data, 10 different models are tested. However, all models' performances are low. To find out the reasons of failure, linear regression is used to analyze the relationship of features with y. Correlation coefficient analysis shows features may not independent with each other.

Introduction

To be successful at college is a popular topic among educators, parents and students. Usually, the common conception about "success" is a good GPA of each "successful" student. Then, more questions come out. Who will usually get high GPA at college? What kinds of element mostly contributing to the high GPA for those successful students? This project studies a training data of UCSC students and try to predict the successful or not on testing data. The purpose of the project is to find the best model to estimate future student who can do "well" (based on GPA) in college.

The data includes training set and test set, which are obtained from Professor David Helmbold's course website. It is admission data for Fall 2013 at UCSC. The training data includes y label (Firstyrcumgpa) and 17 features, such as gender, family incomes, SAT scores, High School GPA, native language et. Al. However, the test data only has 14 features and has no y label. As the data itself has many missing values and training data has more features than test data, so necessary preprocess the data is needed. For training set, extra features are removed to match test data. Finally, the labeled y is classified into 3 classes for further analysis.

After preprocess the data, different models are tested in order to get the best one on test data. More than 10 modes are tested, including NaiveBayes, Logistic regression, SMO, IBK, AdaBoostM1, Bagging, Stacking, tree-LMT, tree-Hoeffding Tree. et al. Under the first treatment of missing data, the best model, based on accuracy on 10-fold cross validation, is meta AttributeSelectedClassifier. It accuracy is 47.03%. Even though it is the best model on training set, its accuracy is still low. Further analysis is to find the reasons for the low accurate learning algorithm. Linear regression is used to analysis the correlation of each feature with y. One

possible reason for the low accuracy of the learning algorithm is the independent features.

Method

1. Tools

To analysis the low accuracy of learning algorithm, this project uses a software named Weka[2], which is developed by University of Waikato, New Zealand. Weka-3-7-12 is the edition.

2. Preprocess the data

Training data:

Removed features: Subjnum, Firststyrunitsforgpa, Firststyeartotcumunits

Feature remained: Firgen, famincome, SATCRDG, SATMATH, SATWRTG, SATTotal, HSGPA, ACTRead, ACTMath, ACTEngWrit, APEScore, FirstLang, HSGPAunweighted

Re-label y : y is defined as Firstyrcumgpa. Instance with missing y is removed from training set. The people with y value in the range of [0, 2.755] is classified into one class. About 963 people in this class. People with y value in the range of (2.755, 3.345] is classified into second class. About 954 people in this class. People with y value in the range of (3.345, 4] is classified into the third class. About 966 people in this class. Option from weka, filters, Unsupervised, attribute, Discretize, useEqualFrequency.

Missing value : missing value of the features is replaced by mean value of each feature respectively. Option from weka, filters, Unsupervised, attribute, ReplaceMissingValues.

Testing data:

Removed features: Subjnum

Feature remained: Firgen, famincome, SATCRDG, SATMATH, SATWRTG, SATTotal, HSGPA, ACTRead, ACTMath, ACTEngWrit, APEScore, FirstLang, HSGPAunweighted

Y value : Test data doesn't have y value. In order to run prediction on weka, y value is added into test data and it is identity to value of HSGPAunweighted for each student. Then classification is done according to this y value. About 100 people with the lowest y value are classified into one class. About 100 people with the highest y value are classified into one class. The rest of 100 people with y value in the middle are classified into one class. Adding y value is done by excel. Classification on y is done by Weka, filters, Unsupervised, attribute, Discretize,

useEqualFrequency.

Missing value : missing value of the feature is replaced by mean value of each feature respectively. Option from filter, Unsupervised, attribute, ReplaceMissingValues.

Notation: weka predicts test data is not relying on y value of test data. Predication is based on model and features of test data. The y value is added into test data only for the purpose of running prediction of weka.

3. Test method on weka

NaiveBayes: weka, classifiers, bayes, NaiveBayes
Logisic regression: weka, classifiers, functions, Logistic
SMO: weka, classifiers, functions, SMO
filterType: Normalize training data
Kernel: RBFKernel
Gamma: 0.01
IBK: weka, classifiers, lazy, IBK
KNN: 9
nearestNeighbourSearchAlgorithm: LinearNNSearch
AdaBoostMI: weka, classifiers, meta, AdaBoostMI
Classifier: HoeffdingTree
leafPredictionStrategy:Naïve Bayes adaptive
AttributeSelectedClassifier: weka, classifiers, meta, AttributeSelectedClassifier
Classifier:LMT
Evaluator: GainRationAttributeEval
Search: Ranker
Bagging: weka, classifiers, meta,Bagging
Classifier: tree, RandomForest
LogitBoost: weka, classifiers, meta, LogitBoost
Classifier: DecisionStump
Stacking: weka, classifiers, meta, Stacking
metaClassifier: DecisionTable
search: BestFirst
Tree: weka, classifiers, trees, LMT

SimpleLinearRegression: weka, classifiers, functions, LinearRegression
y value should be numeric value rather than classes. This method is used to analysis the correlation of each feature with y.

All the method is based on 10-fold cross-validation. To run prediction on test data, one of the methods is chosen to save the model, then load the model and

supplied test data. Finally, Re-evaluate model on current test set. Output the prediction in CSV format.

Results

Performance comparison

Table 1 Best performance of different methods on 10-fold cross-validation

Methods on 10-fold cross-validation	Accuracy (%)
NaiveBayes	44.78
Logistic Regression	46.38
SMO (RBFkernel)	43.32
IBK (kNN=9, LinearNNSearch)	41.86
AttributeSelectedClassifier (LMT)	47.03
Bagging-RandomForest	44.61
AdaBoostM-HoeffdingTree	44.78
LogitBoost-DecisionStump	44.81
Stack-DecisionTable	33.51
Tree-LMT	47.03

Training data are tested on 10 different methods including NaiveBayes, Logistic Regression, SMO, IBK, AttributeSelectedClassifier, Bagging, AdaBoost, Stack, Tree (Bishop 179-218). The accuracy varies from 33.51% to 47.03% on 2883 instances. The average accuracy of the 10 methods is 43.81%. Compared with 33.33% base line of each class, 43.81% accuracy is rather low. The variances of accuracy of all the methods are rather low as well, 0.0015. This means all methods have low performance on the training data. If a method performs well on the training data, this method is a good model for the data. If all the methods have low performance, it can imply that the data itself is hard to study.

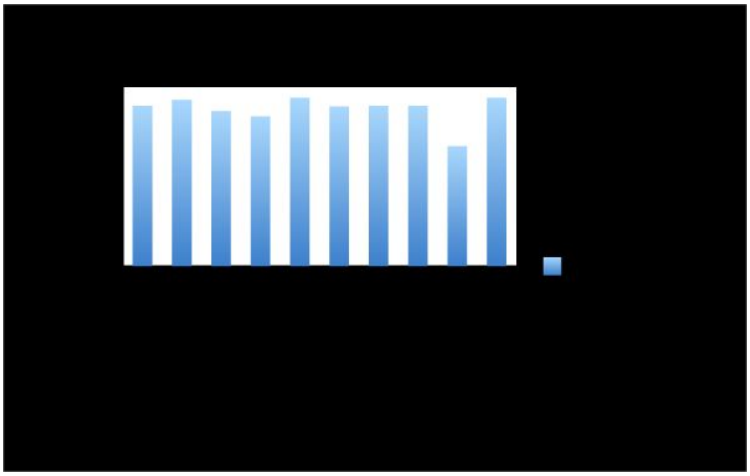


Fig 1 Accuracy of different methods on training data
with 10-fold cross-validation

Fig1 shows how similar of each model's performance on training data. As no model is good enough to continue the original designed experiment, further experiment is to find the reason for the low performance of all test models. In order to analysis the correlation of each feature with y, the simplest method linear regression is used (Trevor 11). When 14 features is analyzed, the final formula is given as below:

Y= 0.2045 * gender
- 0.0003 * Firgen
- 0.0004 * SATCRDG
+ 0.0004 * SATMATH
+ 0.0011 * SATWRTG
+ 0.0001 * SATTotal
+ 0.4487 * HSGPA
+ 0.0175 * ACTEngWrit
+ 0.0198 * APIScore
+ 0.0591 * FirstLang
- 0.302

Feature famincome, ACTRead, ACTMath, HSGPAunweighted are disappearing.

Table 2 Correlation coefficient of features with y

Properties	Correlation coefficient
HSGPA	0.2074
gender	0.088
FirstLang	0.1355
APIScore	0.1687
ACTEngWrit	0.1612
SATWRTG	0.2447
SATCRDG	0.1915
SATMATH	0.1764
Firgen	0.1590
SATTotal	0.2352
famincome	0.0956
ACTRead	0.1182
ACTMath	0.0972
HSGPAunweighted	0.1763
HSGPA	0.2074
HSGPA+gender	0.2199
HSGPA, gender, FirstLang	0.2482
HSGPA, gender, FirstLang, APIScore	0.2914
HSGPA, gender, FirstLang, APIScore, ACTEngWrit	0.3009
HSGPA, gender, FirstLang, APIScore, ACTEngWrit, SATWRTG	0.3311
HSGPA, gender, FirstLang, APIScore, ACTEngWrit, SATWRTG, SATMATH	0.3346
HSGPA, gender, FirstLang, APIScore, ACTEngWrit, SATWRTG, SATMATH, SATCRDG	0.3346
HSGPA, gender, FirstLang, APIScore, ACTEngWrit, SATWRTG, SATMATH, SATCRDG, Firgen	0.3356
HSGPA, gender, FirstLang, APIScore, ACTEngWrit , SATWRTG, SATMATH, SATCRDG, Firgen, SATTotal	0.3352
HSGPA, gender,FirstLang, APIScore, ACTEngWrit , SATWRTG, SATMATH, Firgen, SATCRDG , SATTotal, famincome, ACTRead, ACTMath, HSGPAunweighted	0.3335

It is really interesting that using linear regression, 10 features remain among 14 features. So, further experiment focus on each feature's correlation with y and relationship between features.

Table 2 show each feature has relatively well relationship with y, around 0.1~0.2. It means each feature studied along with y, the correlation coefficient of this feature with y is fairly good, around 0.1 ~ 0.2. However, when features are studied together by linear regression, the total correlation coefficient is rather low. Feature HSGPA, gender, FirstLang, APIScore have 0.2074, 0.088, 0.1355, 0.1687

correlation coefficient (cc) individually. When studying them together, the total cc is 0.2914, which is much lower than the sum of each individual cc. In another words, 0.2914 is much lower than 0.2074+0.088+0.1355+0.1687. Further more, if feature ACTEngWrit is selected along with feature HSGPA, gender, FirstLang, APIScore, the total cc increases to 0.3009. There is only 0.0095 different between four features and five features considering feature ACTEngWrit itself has cc 0.1612 individually. When more features studied together, this situation is more obvious. When feature HSGPA, gender, FirstLang, APIScore, ACTEngWrit, SATWRTG, SATMATH, SATCRDG, Firgen exist together, the total cc reaches the maximum. Continue adding feature for studying cause the total cc unchanged or decreased. This may imply that there are certain correlations among features or features are not independent with each other. This kind of relation makes learning hard. So, the 14 features are not equally important for learning. This is probably the reason when 14 features studied together, only 10 features show at linear regression formula.

Discussion

Accurately estimate the successful students in future are the original purpose of the study. However, low performance of 10 models lead the study focus on some possible reasons for the failure. Correlation analysis shows each feature may not independent with each other. This relationship of the features adds difficulty to learn the pattern of the data.

So, linear regression analysis show the features are not independent with each other. The further question is to what extend one feature can be covered by other features and how many features can be covered by others? For higher dimension feature space, will it happen again? Further research can test models that don't require iid of the data. Maybe more iterations can be a good way as well.

This project help me understand the learning methods, such as regression, Naive Bayes, Tree methods. Especially when I try to find the reason for failure, I read the linear regression carefully. Furthermore, I try to think of method with high feature dimension spaces. Classnotes and slices are read many times when I try to figure out the reason. I also search the methods AttributeSelectedClassifie and LogitBoost-DecisionStump on line to understand them. Actually, more experiments are done for this project. Owing to limited time, I couldn't represent all of them in this project. This project leads me to think about all the methods I learn (their benefit, limitation) and how to use them.

Second Approach:

Method

This is a scikit-learn[1] based approach, for binary classification of the dataset.
Scikit-learn is a machine learning library for Python.

Tools

- Python 2.7
- scikit-learn
    - KMeans
    - train_test_split
    - GaussianNB
    - Imputer
    - DummyClassifier
    - confusion_matrix
    - decomposition.PCA
    - GridSearchCV
    - svm.SVC
    - Pipeline
    - MinMaxScalar
    - KNeighborsClassifier
- NumPy
- Excel

Preprocessing

The data comes as CSV, which is a format where each value is separated by a comma. This is easy to read into the program, but the contents of the data cannot be used in a raw form. There are missing values, some features are words, others are numbers, and it generally cannot be processed in this state. So there is a preprocessing step. There are subtle but important differences in how training data is processed, compared to how unclassified data is trained, so the two processing will be described separately.

Training Data

Data is read in from the file, and defined functions convert word-based features to use numerical values. Specifically, *gender* and *FirstLang* (whether or not the student's first language was English) each have three words available to describe a student, and in this step those words are mapped to the numbers 0, 1, 2. Now all the data is stored in an array, and all of the values are numeric.

The next step is to extract the specific instances (students), and features we want to work with. First, only instances with data for *Firstyrcumgpa* (the student's GPA at the end of their first year) are selected. From those instances, we select the instances which have values for all the features in our selected feature set, and we discard data for all features not in the selected set. So we have selected specific features, and only instances which have data for all of those features.

At this point we train an Imputer algorithm. It collects the median value for each feature, and will be used to fill in missing data for our test instances.

The selected data is then scaled, excluding the *Firstyrcumgpa* data. Each feature is scaled individually, fitting into the range [0,1]. A scaling algorithm is trained on this data, collecting the min and max of each feature, so that test instances can be put on a scale familiar to the classification algorithm.

Now we have a scaled, selected set of instances, an unscaled set of GPAs which correspond with those instances, and two algorithms which have been adjusted for this dataset.

The final preprocessing step is to separate this data into two groups, a training set and a development set. This is done with scikit-learn's train_test_split function. The data is partitioned differently depending on intended use.

Test Data

The data is read into the program and changed to all numeric values. Because this data must be classified, no instances can be tossed out. Once we isolate the selected feature set, it is processed with the trained Imputer object, which fills in missing values, and then the data is scaled by the MinMaxScaler.

Labelling

The training data is labeled according to *Firstyrcumpga*. If the GPA is greater than 3.0, the instance is labeled with a "1". GPAs equal to or less than 3.0 are labeled with a "0". This results in a balanced labeling of the training dataset, where half are "1", and the other half are "0".

Model Fitting

Three classification algorithms were explored in this part of the project: Gaussian Naive Bayes, K-Nearest Neighbors, and Support Vector Machine (SVM) with the radial basis function (RBF) kernel.

Gaussian Naive Bayes was fit to the training set without any special parameters.

K-Nearest Neighbors was fit using n_neighbors = 9.

Parameters for the SVM were found using an exhaustive grid searches on two different sets of features. Each parameter was tested with 20 different values, and the combinations were evaluated with 5-fold cross-validation. *C* values were chosen from [10^0.01, 10^6] and *gamma* values were chosen from [10^-5, 10^5].

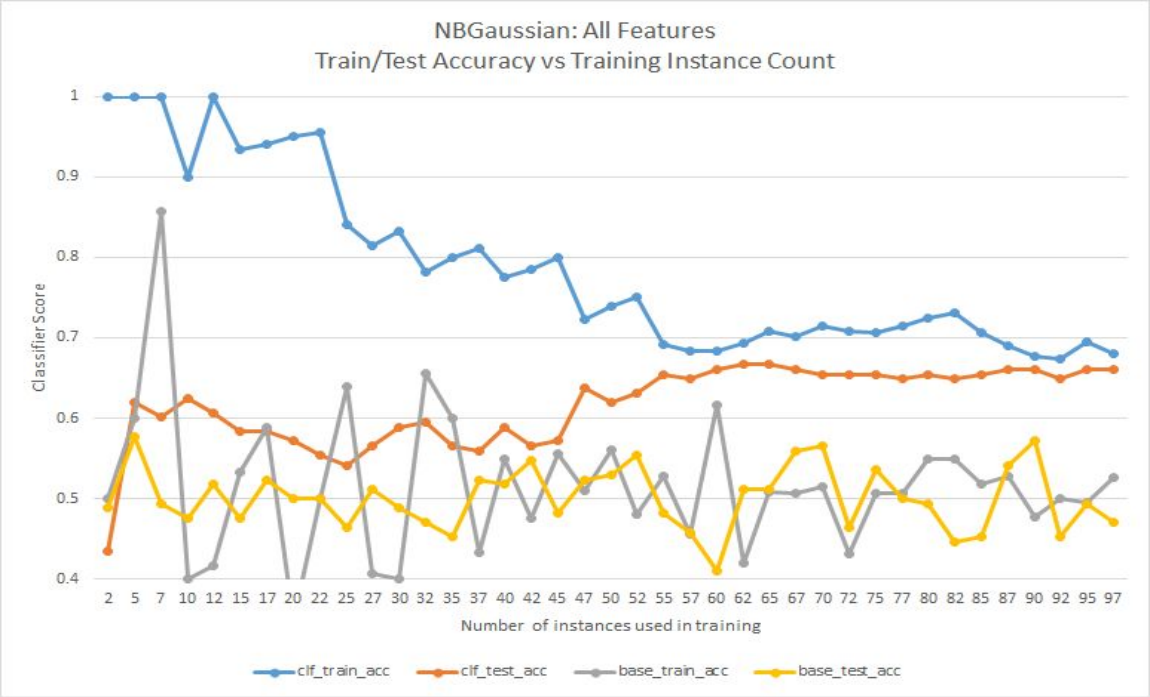
The grid search for one feature set (SAT Subject Tests and High School GPA) was run on 50% of the training data and took 6.75 hours. The optimal parameters are *C*=239302.57311, and *gamma*=0.00428133.

The other grid search was done with the entire feature set on 25% of the training data and took 5.5 hours. The optimal parameters are *C*=22.25197, and *gamma*=0.0072813.

Evaluation

The algorithms were evaluated with a comparison against a baseline value of accuracy. The training labeled data is evenly divided, so the baseline is 50% accuracy. That is, if instances were uniformly, randomly classified, about 50% accuracy is expected. The scikit-learn classification algorithm DummyClassifier does exactly this, and so it was used to establish a baseline for comparison.

Below is a chart demonstrating the evaluation of Naive Bayes as it works on the full feature set:



The algorithm's accuracy when classifying the data it was trained on is compared to its accuracy when classifying never-before-seen data. Here, 25% of the training instances are held out to use as a test set. The two accuracies converge when 60 instances are used to train the classifier, and the values after the point of convergence are what we used to decide how well a classifier performed.

Confusion matrices, as generated by scikit-learn, were also used. A confusion matrix is useful for seeing if an algorithm has a tendency to assign one label more often than the other.

Classification

The process of classification was left entirely up to scikit-learn. A trained classifier object has a *predict* function. The preprocessed test instances are passed as arguments to the *predict* function of a classifier object, and classes are returned in an array.

Results

The following table contains accuracies from algorithms trained on 75% of the training data, scored by classifying the held-out 25%. Please, look in the appendix for the corresponding charts.

Algorithm	***Feature Set***	***Accuracy (%)***
Gaussian Naive Bayes	All Features	65

	SAT Subject Tests, and High School GPA	61
	Non-academic	60
K-Nearest Neighbors (KNN)	All Features	60
	SAT Subject Tests, and High School GPA	60
	Non-academic	55
SVM (RBF kernel)	All Features	65
	SAT Subject Tests, and High School GPA	62
	Non-academic	56

Discussion

Results observations

On the full feature set, Gaussian Naive Bayes performed similarly to SVM and was much faster to train. However, looking at the charts, Naive Bayes does not indicate an issue with variance on the non-academic feature set, when both of the other algorithms behave as though that feature set has the most variance.

All three of the algorithms exhibit the same bias, and perform generally similarly. KNN shows issues with variance for all three feature sets.

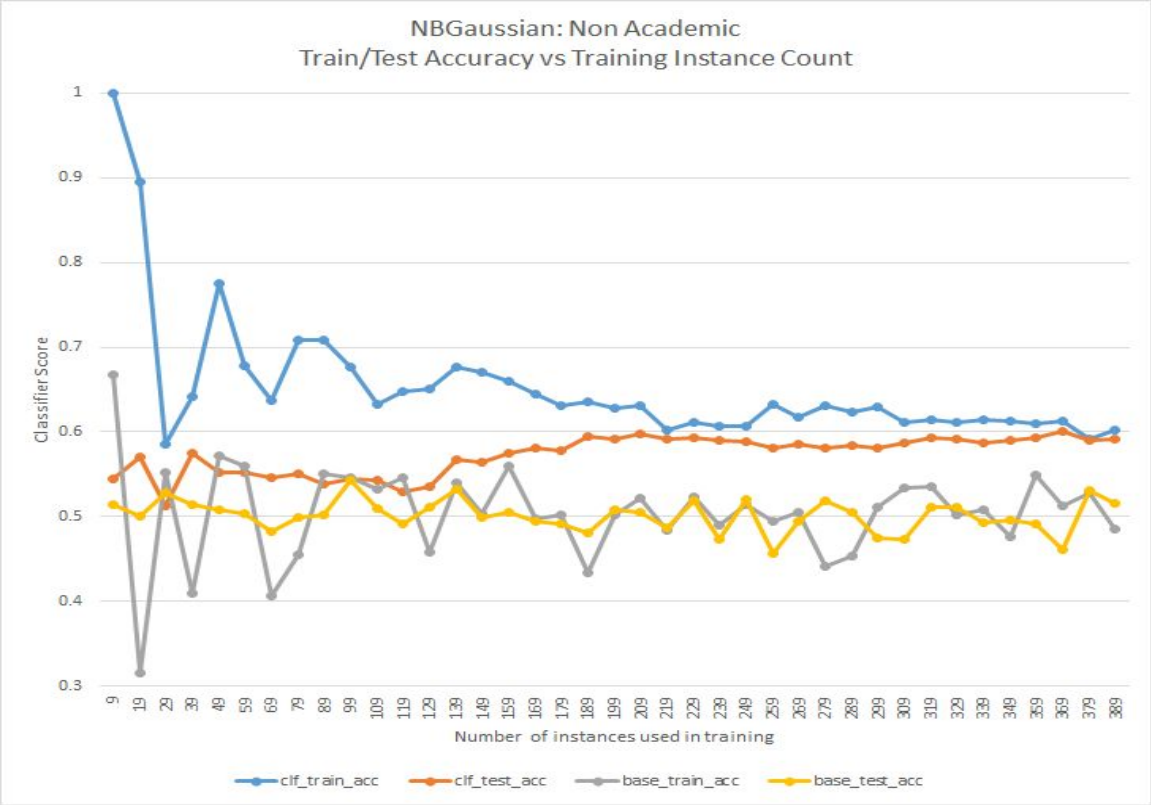
What to improve…

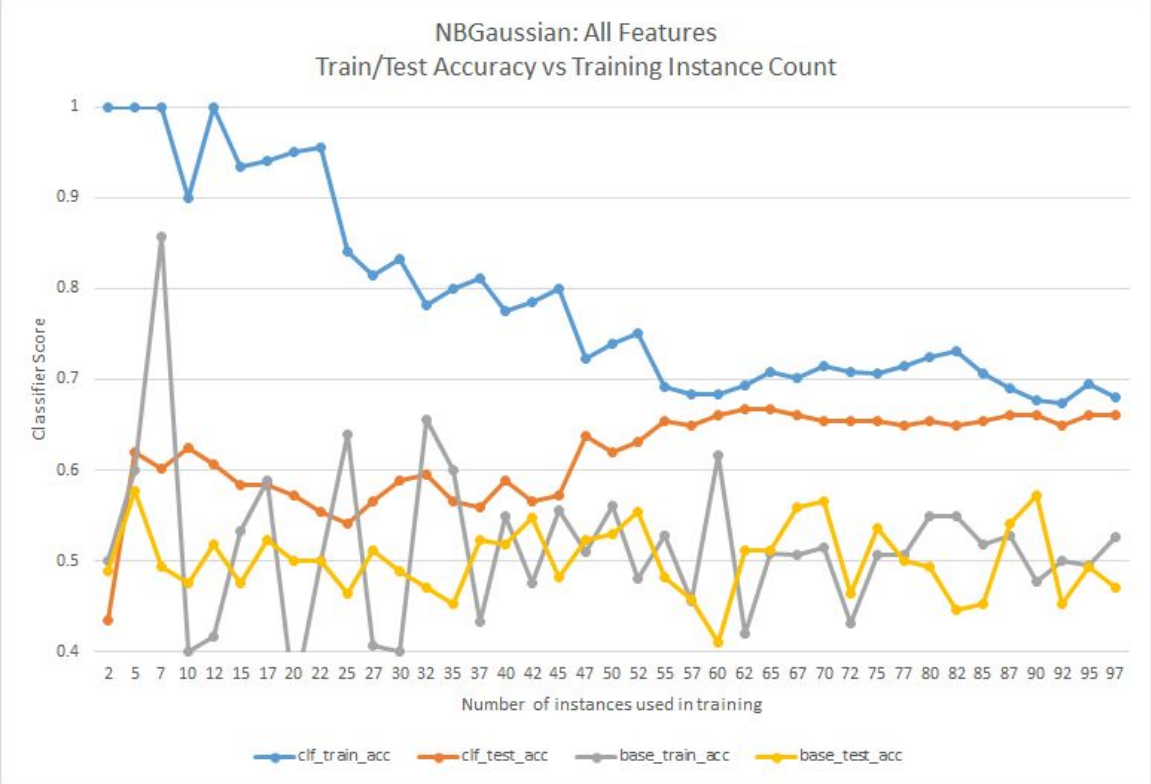
A strong bias could be the cause of all three algorithms showing similar performance. Some error was likely introduced by the decision to remove instances with incomplete data. Although it was not tested here, inserting median data is also likely to influence the bias (and the variance).

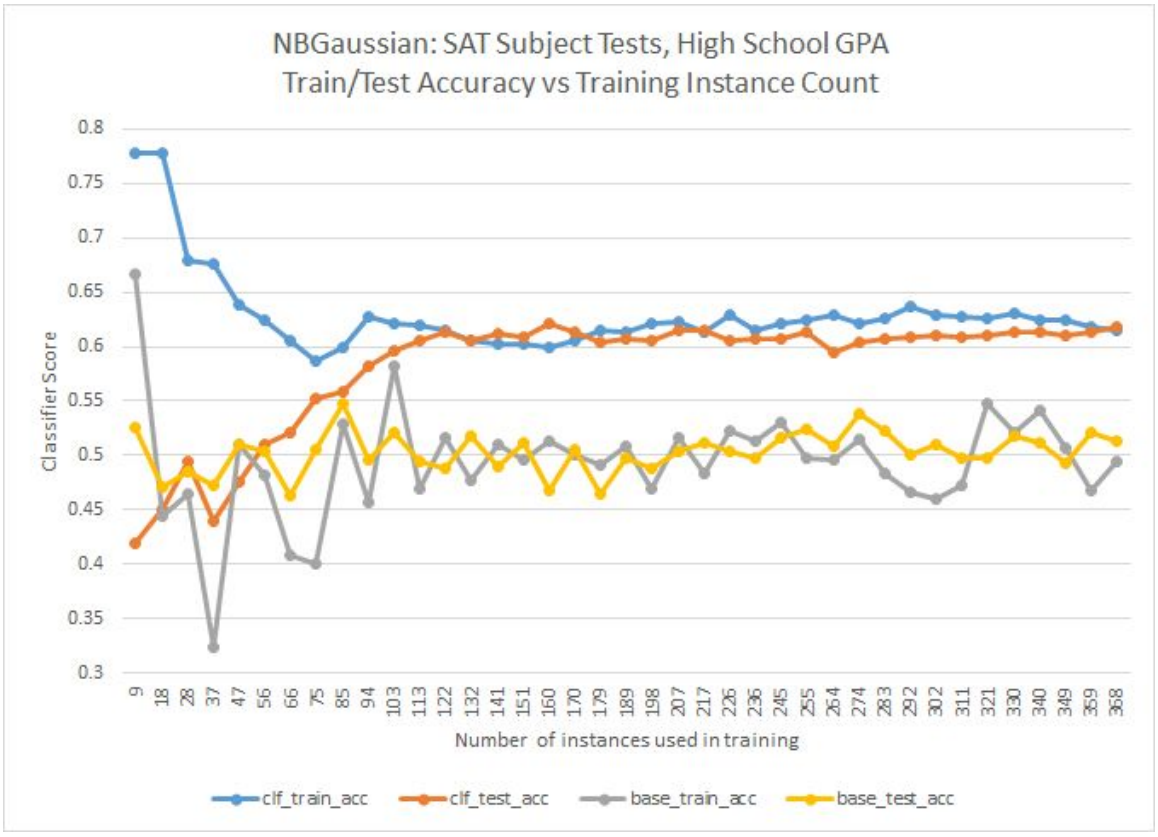
Scaling each feature independently is certainly a cause of error. An example of this can be seen in one specific instance. In the dataset there is a student whose family income is $7,620, and High School GPA is 1.76, but their *Firstyrcumgpa* is 3.0. It may not be unreasonable to imagine that this student's family income influenced their high school GPA. If the High School GPAs of students from different income brackets were scaled separately, this student's 1.76 might not be put at the bottom of the scale.

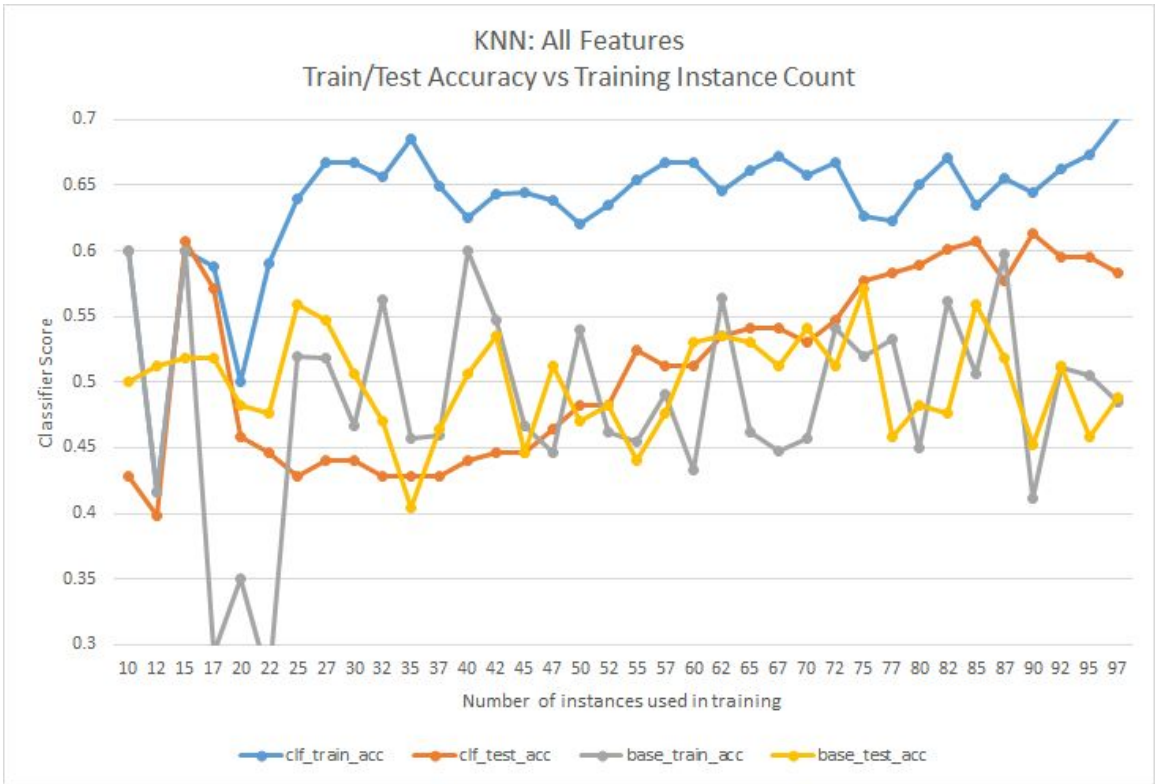
Variance is another issue with this dataset. The exact same thorough grid search can be run twice on the same data and yield dramatically different results. It is likely that the features in the dataset do not account for everything which influences a student's success in their first year of college.

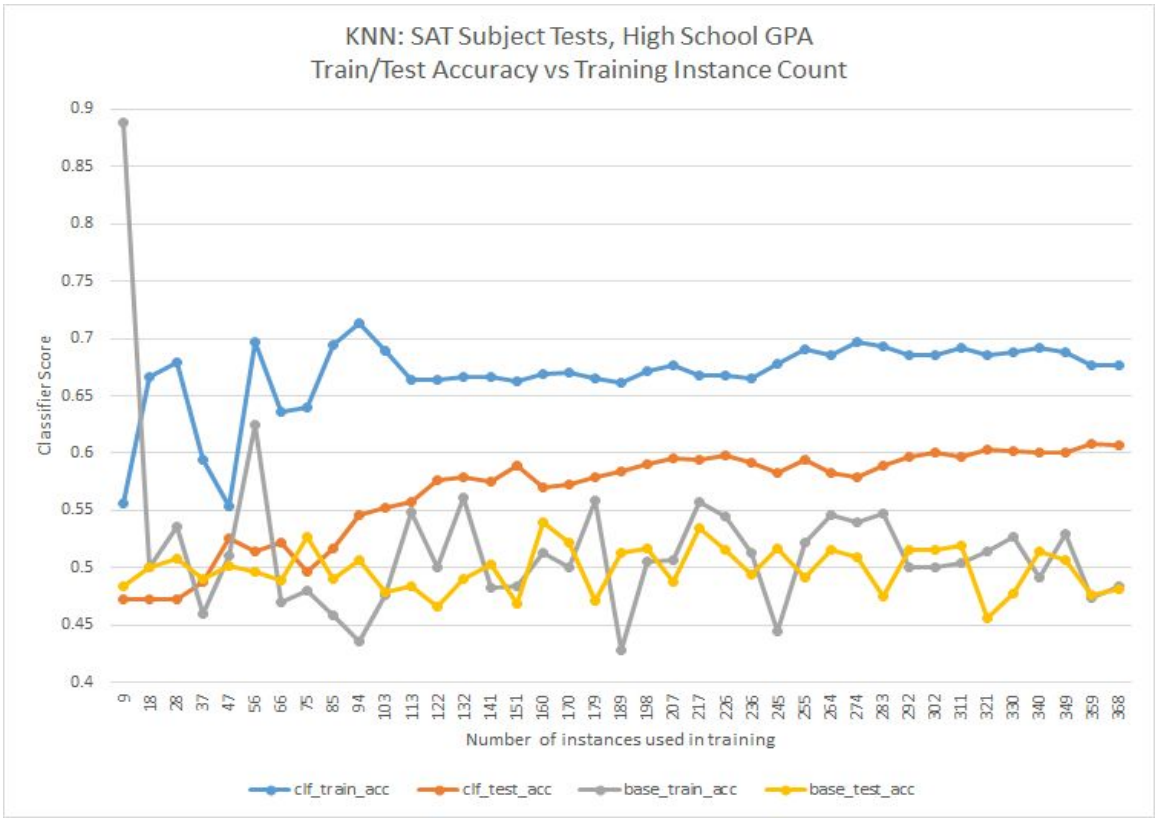
Appendix

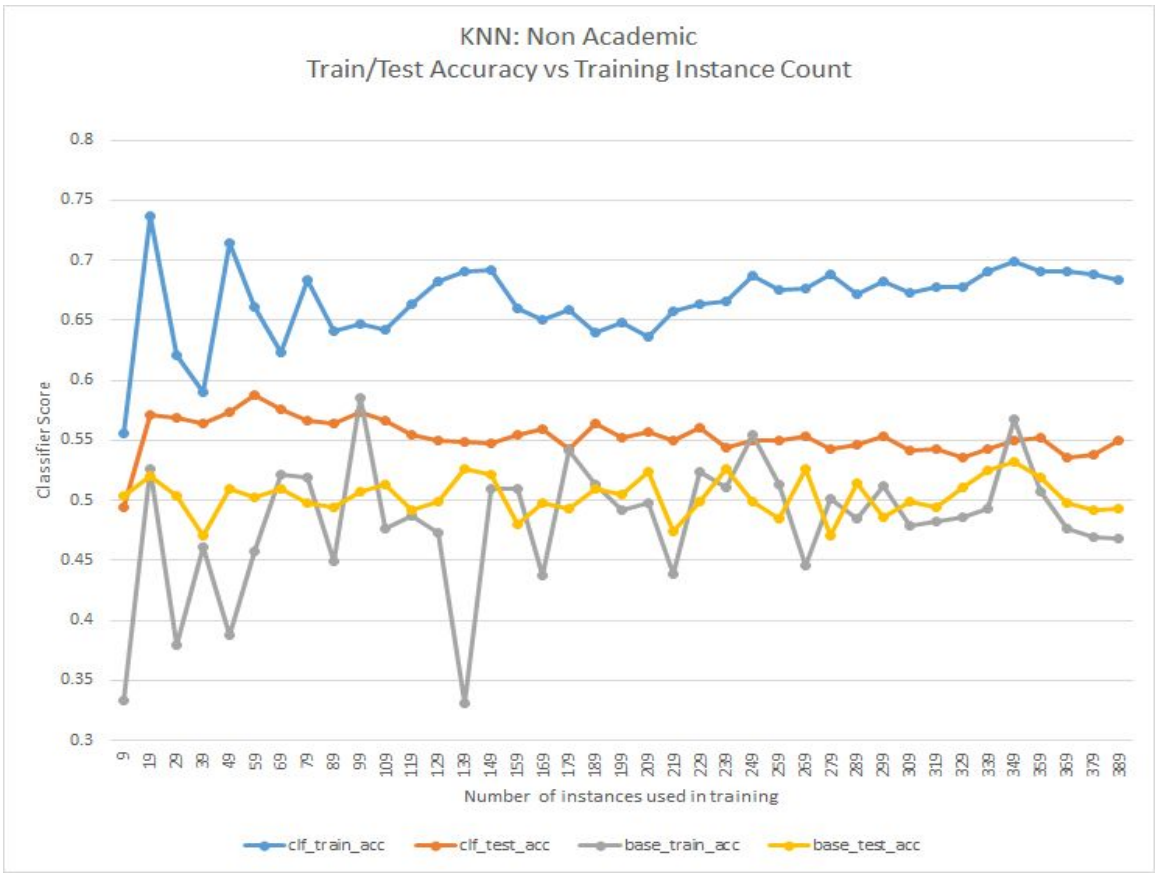

NBGaussian: Non Academic
Train/Test Accuracy vs Training Instance Count
Classifier Score
Number of instances used in training
clf_train_acc
clf_test_acc
base_train_acc
base_test_acc

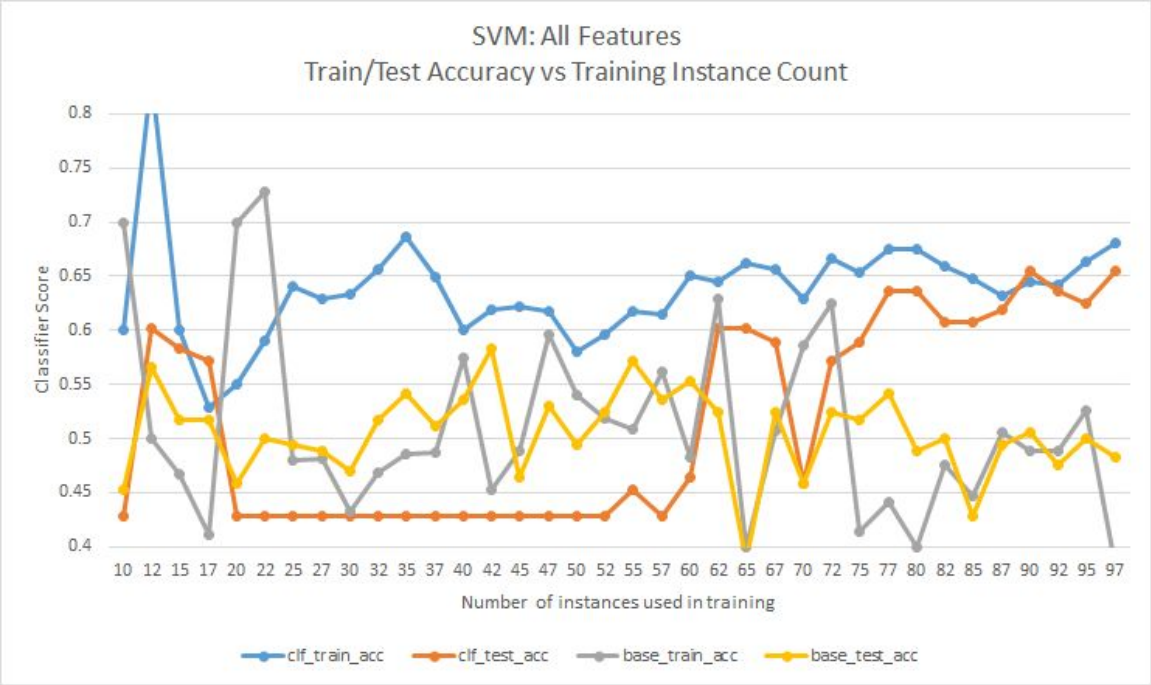

NBGaussian: All Features
Train/Test Accuracy vs Training Instance Count
Classifier Score
Number of instances used in training
clf_train_acc
clf_test_acc
base_train_acc
base_test_acc

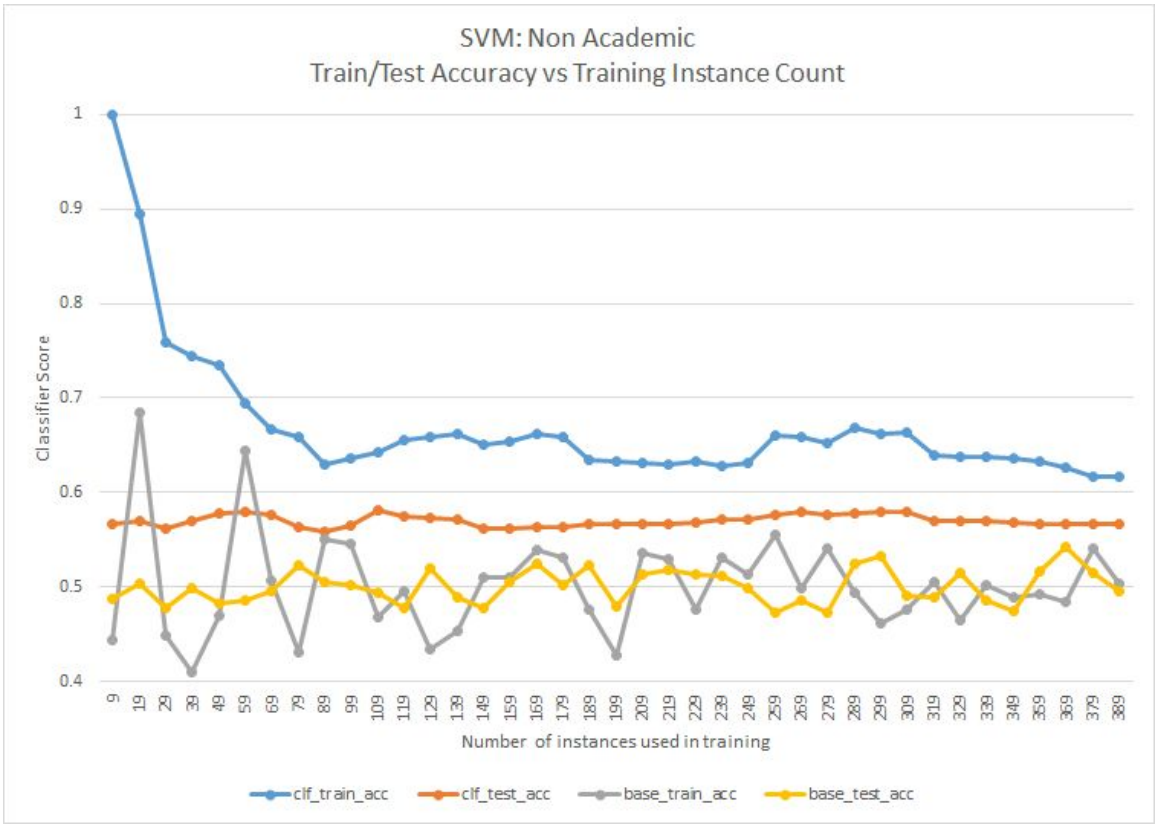

NBGaussian: SAT Subject Tests, High School GPA
Train/Test Accuracy vs Training Instance Count
Classifier Score
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
9 18 28 37 47 56 66 75 85 94 103 113 122 132 141 151 160 170 179 189 198 207 217 226 236 245 255 264 274 283 292 302 311 321 330 340 349 359 368
Number of instances used in training
clf_train_acc
clf_test_acc
base_train_acc
base_test_acc

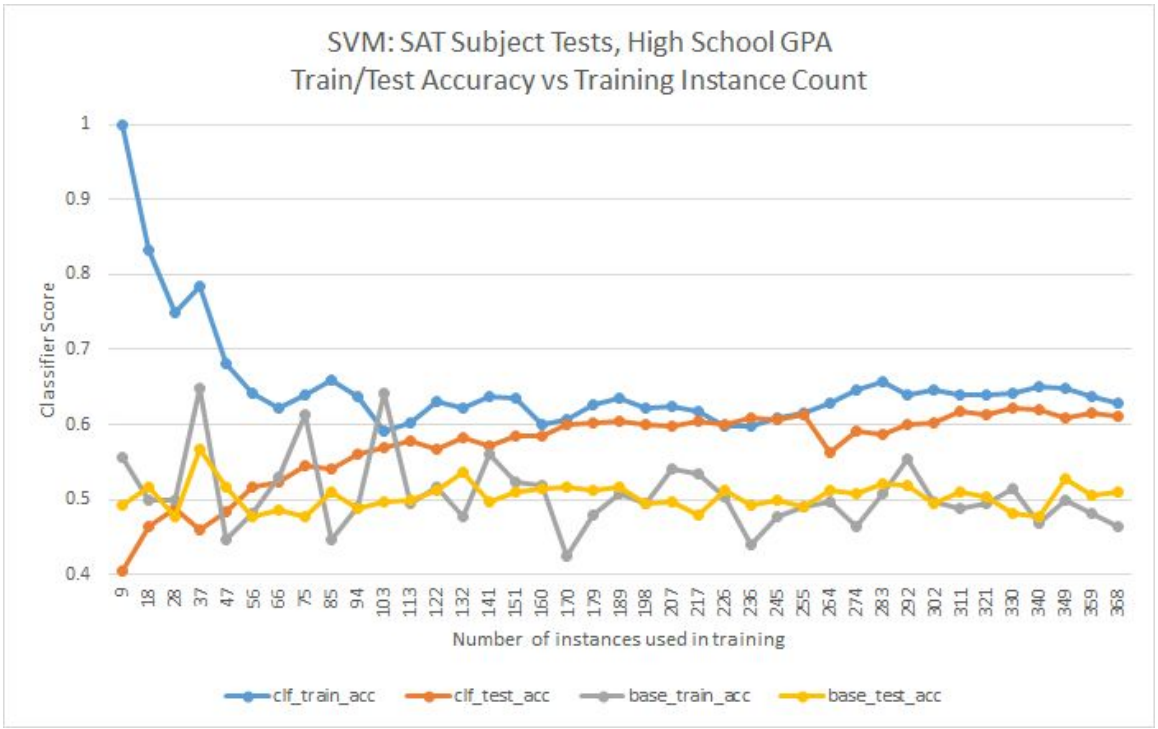

KNN: All Features
Train/Test Accuracy vs Training Instance Count
Classifier Score
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
10 12 15 17 20 22 25 27 30 32 35 37 40 42 45 47 50 52 55 57 60 62 65 67 70 72 75 77 80 82 85 87 90 92 95 97
Number of instances used in training
clf_train_acc
clf_test_acc
base_train_acc
base_test_acc


KNN: SAT Subject Tests, High School GPA
Train/Test Accuracy vs Training Instance Count
Classifier Score
Number of instances used in training
clf_train_acc
clf_test_acc
base_train_acc
base_test_acc

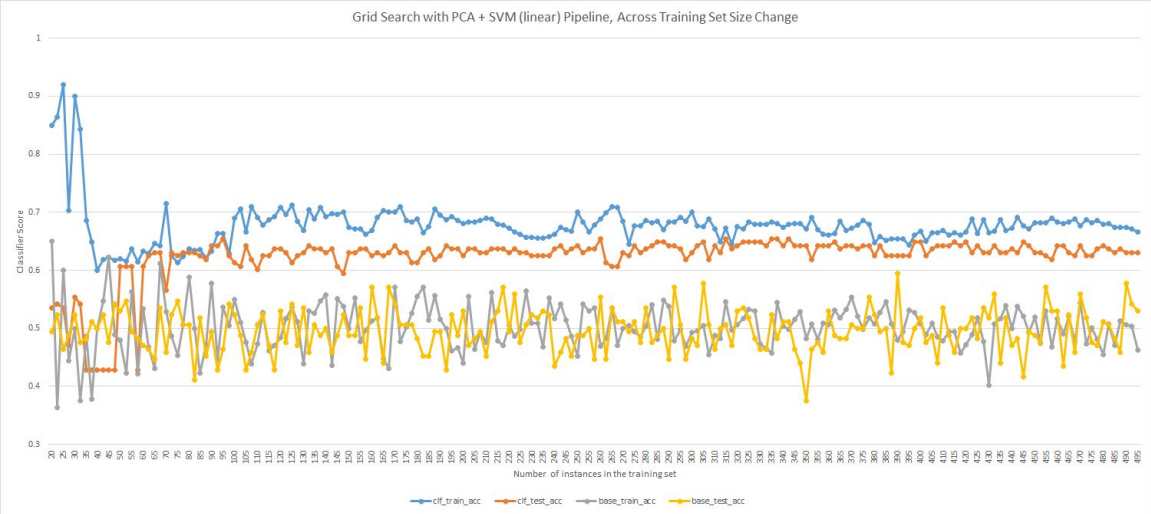

KNN: Non Academic
Train/Test Accuracy vs Training Instance Count
Classifier Score
Number of instances used in training
clf_train_acc
clf_test_acc
base_train_acc
base_test_acc

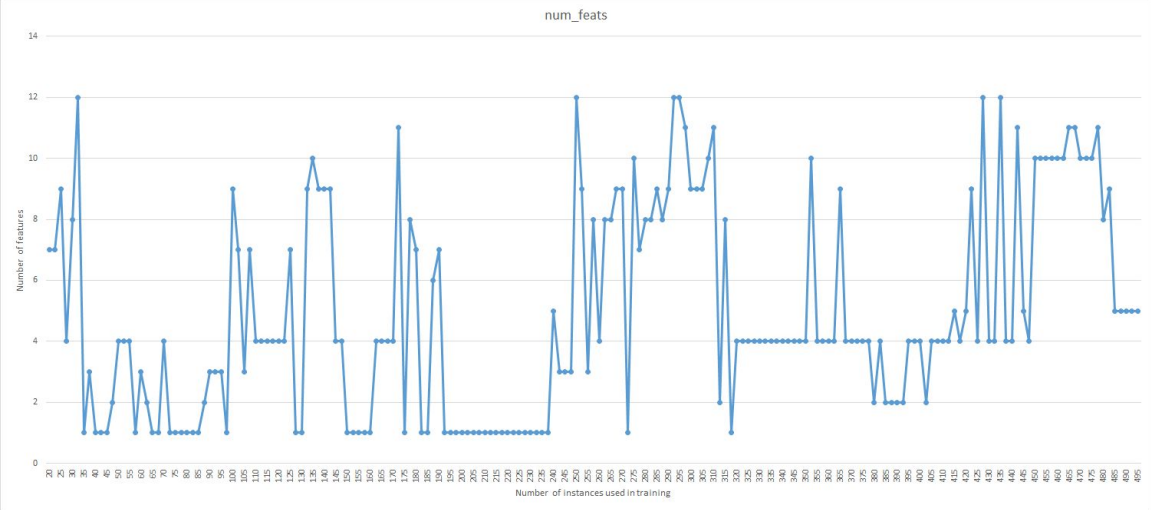

SVM: All Features
Train/Test Accuracy vs Training Instance Count
Classifier Score
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
10 12 15 17 20 22 25 27 30 32 35 37 40 42 45 47 50 52 55 57 60 62 65 67 70 72 75 77 80 82 85 87 90 92 95 97
Number of instances used in training
clf_train_acc
clf_test_acc
base_train_acc
base_test_acc

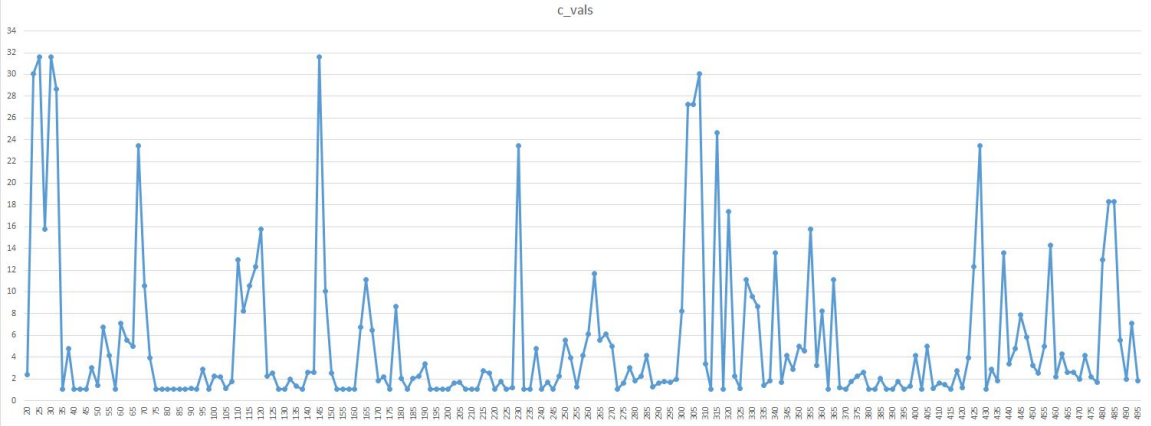

SVM: Non Academic
Train/Test Accuracy vs Training Instance Count
Classifier Score
Number of instances used in training
clf_train_acc
clf_test_acc
base_train_acc
base_test_acc


SVM: SAT Subject Tests, High School GPA
Train/Test Accuracy vs Training Instance Count
Classifier Score
Number of instances used in training
clf_train_acc
clf_test_acc
base_train_acc
base_test_acc

Extra charts (not referenced in report):







Bibliography

1. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
2. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.
3. Fortmann-Roe, Scott. "Bias and Variance." Understanding the Bias-Variance Tradeoff. 2012. Web. 10 June 2015.
4. Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer. 2006. Print.
5. Trevor Hastie, Robert Tibshirani, Jerome Friedman. *The Elements of Statistical Learning-Data mining, inference, and prediction*. Springer. 2001. Print.