**Classifying children with Specific Language Impairment (SLI)**

Shane Sweeney

Table of Contents

I. Introduction
II. Data Mining Task
III. Technical Approach
IV. Evaluation Methodology
V. Results and Discussion
VI. Lessons Learned
VII. References

## I. Introduction

The dataset I have chosen to work with comes from Kaggle [1]. The dataset is a collection of 3 different datasets that involved different attributes from interviews with children age ranging from 4 to 12. The interviews were conducted to try and identify if a child had Specific Language Impairment (SLI) or not, which would be classified as typically developing (TD).

SLI affects around 7% of 5-year-old children. It is a lack of language ability compared to similar aged children, but not due to direct mental or physical disabilities. Diagnosing SLI in children can be a lengthy process so if there could be a way to diagnose children using machine learning, this could be useful for pediatricians.

I am personally interested in this subject as my younger sister had trouble speaking and mispronounced words quite often when she was around 6. It wasn't until a few years later that my parents realized she needed tubes for her ears. She had been pronouncing words incorrectly because she was not hearing clearly.

I set out to find if a support-vector machine (SVM) model could be applied to the dataset and if a linear or radial kernel would be more appropriate for classifying a child as SLI or TD based on the available attributes. My approach involved using training and testing data to train and test the SVM models. My challenge was not having much experience with SVM and if I would be able to appropriately apply the model.

In the following sections I will discuss the issues that arose during this project. Briefly, my results were that a linear kernel SVM model with cost 5 had an AUC of 0.49. A radial kernel SVM model with cost 10 and gamma 2 had an AUC of 0.5. I then attempted to use a randomForest model which resulted in an AUC of 0.58 which was much better than the other SVM AUC's but still performed poorly. I must also note that I attempted to set seeds where I wanted to reproduce the same results over and over, but my program was still outputting slightly varying results at times. Also, where I wouldn't expect variation after training my SVM models, I would get different test accuracies at times. I also had to learn about the Synthetic Minority Oversampling Technique (SMOTE) for this project as my dataset had imbalanced class distributions.

## II. Data Mining Task

The data mining task I set out to investigate was if a SVM model could be used to classify a child as having SLI or not and if a SVM model with a linear or radial kernel was more accurate. I will break the samples from the dataset into training and testing data. The input data for creating my SVM model will be the training set. The training set and test set will come from a .csv file provided from Kaggle. The samples are in the form of rows with different attributes for the columns:

| age | Group | Child_TNW | Child_TNS | Examiner_TNW |
|---|---|---|---|---|
| 165 | SLI | 287 | 36 | 4 |

(Note: The Kaggle[1] dataset provides more details for each attribute)

My output for my data mining approach will be the creation of an SVM model and also a confusion matrix for training and test accuracy as well as an ROC curve and the AUC. The label for classification will come from the “Group” column which lists the child as “SLI” or “TD”.

The main challenge for this task is appropriately using an SVM package from the CRAN repository in R. The other challenge will be if I cannot get an SVM model to work appropriately what other model might I use? Finally, while conducting this project I noticed I had a class imbalance with SLI samples only representing 23% of the samples. This was an issue of imbalanced classification [2].

## III. Technical Approach

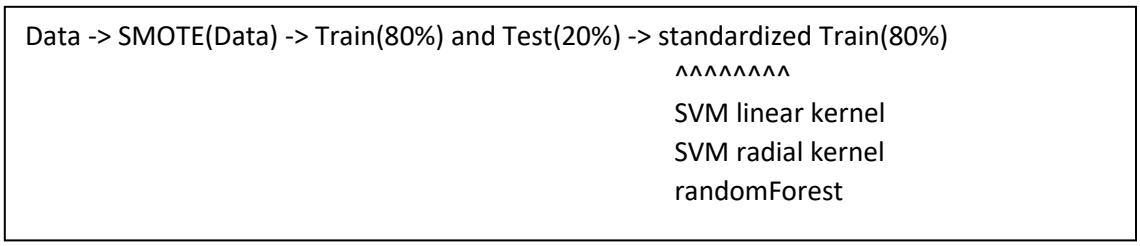


**Figure 1** Workflow diagram for preparing data for training and testing datasets

packages used for models and class balancing

I used the smote() function from the performanceEstimation package in the CRAN repository to address the class imbalance issue.
I used the svm() function from the e1071 package in the CRAN repository to create my SVM models.
I used the randomForest() method in the randomForest package in the CRAN repository for my randomForest model.

Data preparation

I first uploaded the .csv file to a dataframe in R. I used the .csv file on Kaggle that already had highly correlated attributes removed. I then removed three attributes: filename, sex and corpus. Filename was the same for all samples and doesn't add to the study. Sex was NA for some samples so I removed it. Corpus just mentions which study the data was from which would not be necessary for creating my models. I made sure to make the "group" column a factor for "SLI" and "TD" so SVM would work appropriately.

Imbalanced Classification

I initially moved right along with the dataset I had created from the steps above and created a training and testing set. I created two SVM models with linear and radial kernels and found that the models were very inaccurate. I knew there was an issue with the "SLI" samples being a minority and only representing 23% of the data. I didn't know how to correct the issue though after some initial trials. I then read in the Kaggle dataset's commentary that the original user employed a method known as SMOTE. This led me to reading up on the SMOTE method.

SMOTE is a method specifically for scenarios like mine where the minority class is highly under represented [2]. Most machine learning models assume there is an even amount of samples per class when it comes to classification. If one of the classes is severely misrepresented, the minority class, the model will undervalue it and not report accurately. SMOTE creates synthetic samples from the minority class by creating new samples that are in the same feature space of the original samples. It does this by looking at for example, the k=5 nearest neighbors of a sample, then creating a new synthetic sample that could have occurred. This is known as over-sampling. SMOTE also under-samples from the majority class.

In this manner SMOTE can artificially make two classes get closer to a 50/50 ratio of representation.

Technical approach

My technical approach or workflow is represented in figure 1. After tuning the smote() method on the prepared dataset by selecting different perc.over/perc.under combinations I managed to get the "SLI" samples to 57% of the total dataset and the "TD" samples to 43% of the total dataset (which is much closer to 50/50 compared to 23% to 77% before using SMOTE) I then created a training and testing dataset. I split the data 80% to training and 20% to testing. I then standardized the training set as SVM models can be affected by the scale of the features. I then tuned the SVM model for linear and radial kernels by trying different costs for both and different gammas for the radial kernel specifically. I then created SVM models by using the svm() method and training the model on the training dataset. I then tested the SVM models by comparing it's predicted classifications to the test set's "group" column. I looked at confusion matrices by using the table() function. I also looked at ROC curves and the AUC.

Addressing challenges

Regarding using the SVM method appropriately I made sure to look at examples of its use in R and also read through the package documentation. As I will discuss in the Results and Discussion section, my SVM models were not producing helpful classifications. I looked over the remarks on the Kaggle dataset and I read that randomForest was a model that could work well. I decided to use the randomForest() method and followed the same technical approach, except when tuning I trialed different number of trees and the maximum number of nodes (which controls the number of leaf nodes for the trees). Regarding the issue of class imbalance I employed the smote() method with the reasons mentioned in the imbalanced classification section above.

## IV. Evaluation Methodology

As mentioned in the introduction and in the data mining task sections my dataset is a 238.7 kb .csv file with 1,163 rows and 50 columns. I downloaded the dataset from the aforementioned Kaggle[1] website. The main challenge with this data has to deal with the class imbalance which I addressed in my technical approach section.

The metrics I used to evaluate the SVM and randomForest models were a confusion matrix, ROC curve and the AUC. I used the confusion matrix to compare the different models' predicted classifications to the test set classification labels and looked at the prediction accuracy of the model by taking the total correctly predicted labels divided by all labels predicted. I also read a bit about what evaluation metrics are most appropriate for imbalanced classification problems and read that AUC is a better evaluation metric than prediction accuracy [3] even though I have still reported the prediction accuracy. An AUC of 0.5 is a classifier that has no skill and an AUC of 1 is a perfect classifier model. AUC values from 0.6-0.7 are considered poor, 0.7-0.8 is fair, 0.8-0.9 is good.

## V. Results and Discussion

After using smote() on the data and creating training and test sets and standardizing the data, I created two SVM models: one with a linear kernel and one with a radial kernel.

| | Linear SVM | Radial SVM | randomForest |
|---|---|---|---|
| **Training accuracy** | 81.6% | 100% | 89.36% |
| **Testing accuracy** | 61.5% | 61.2% | 53% |
| **AUC** | 0.49 | 0.5 | 0.58 |

**Figure 2** training and testing accuracies as well as AUC values

As figure 2 shows the AUC values I was getting for the linear and radial SVM models were basically telling me I was creating models that had no predictability, as any value close to 0.5 is considered a no skill model.

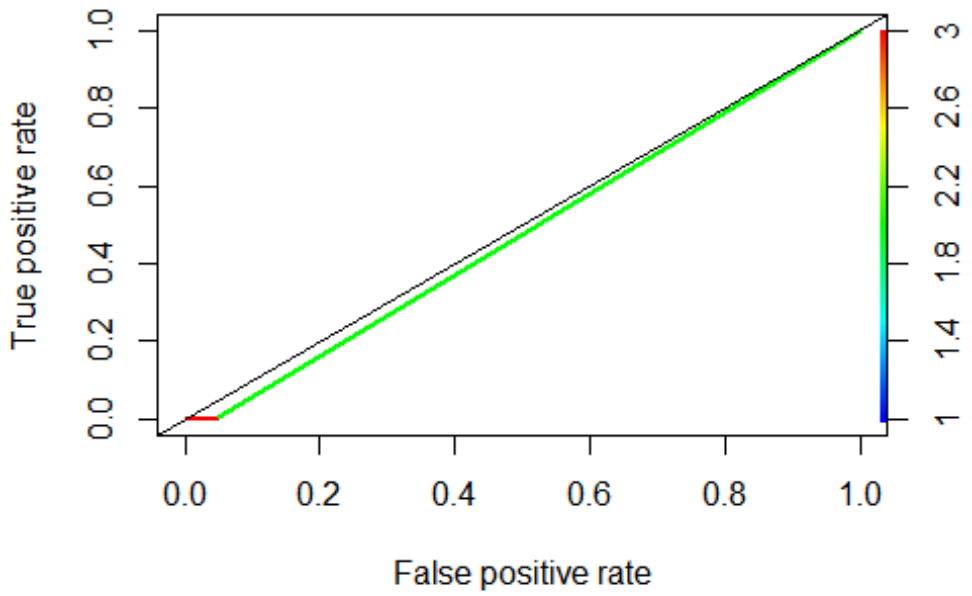


**Figure 3** ROC curve for linear SVM model

As mentioned in my technical approach section I read in the comments of the Kaggle dataset that a randomForest model with the use of SMOTE could be an accurate classifier. Using the randomForest() method and the same procedure to the SVM models I was able to get an AUC value that was closer to 0.6 which at best is a “poor” classifier, but it was much more successful compared to my SVM models.

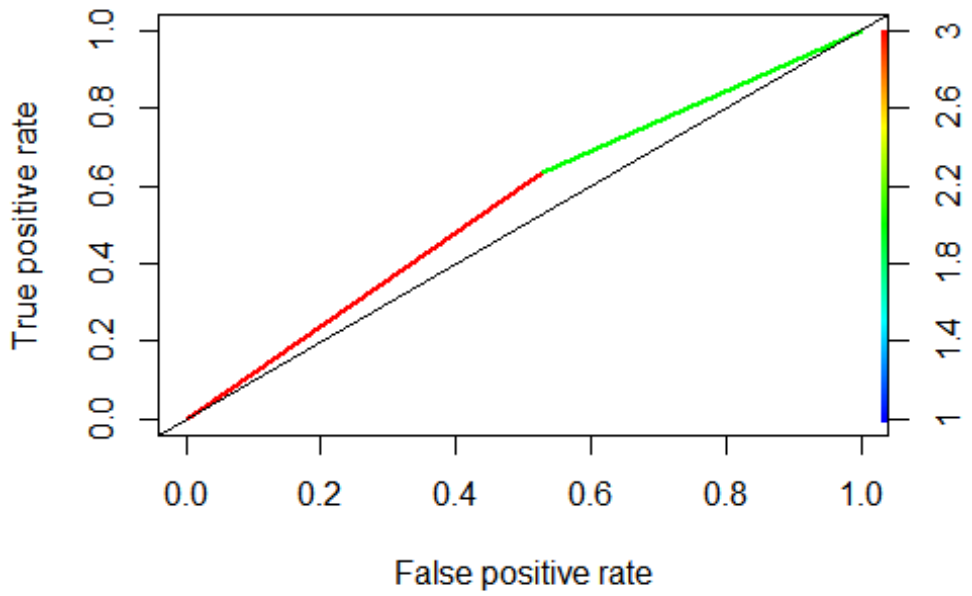


**Figure 4** ROC curve for randomForest model

Comparing figure 4 to figure 3 it can be seen how the ROC curve for the randomForest model would produce a greater AUC value.

What worked for this study was my randomForest model. It was not a good model, but compared to the results of AUC numbers near 0.5 for the SVM models, the randomForest model at least was able to make some predictions accurately. I believe this model worked better as it was able to get averages of multiple trees which by nature could produce some classification predictability consistently if there are actual differences between the SLI and TD classes. This shows to me that my data was not separable by a linear or radial boundary line.

What didn't work was my SVM models. I tried tuning the models with their cost values and for the radial kernel SVM I also trialed different gamma values. I was getting a runtime that would never finish sometimes with the radial SVM model so I had to manually trial different tuning combinations. I believe the SVM models did not work well because as already mentioned above, I do not think the data could be separated by a linear or radial boundary line. This could be due to my use of the SMOTE function which did improve my results but my novice understanding of it could be an issue. The dataset itself could also not be perfect for any model to separate it with much success. Although I'm assuming there is some user error on my part as the comments on the Kaggle dataset say the creator was able to get a good ROC curve using Neural Networks. I attempted to use a Neural Network as a quick trial but still could not get an accurate classifier.

Special note: I was having issues reproducing my results in my rmarkdown file. I was using the set.seed() function which should save results so they do not vary on each run, but I was still getting varying results. I have reported my findings in this report based on my initial running of my script. When my code is run there will be some variation. I've looked it up and this could be due to different version of R between my code and the packages I'm using.

## VI. Lessons Learned

To answer the question of this study if a SVM model could be successful in identifying a child with SLI or not, I believe the answer is yes. My SVM models were not successful based on what I brought up in the results and discussion section, but from reading the comments on the Kaggle dataset, I know a SVM model with SMOTE can be a good classifier. I believe there is user error on my part and since I am a novice with the SMOTE method, there may be a misunderstanding I have on its use that is causing my SVM models to not work as well.

I feel like I set out to learn more about SVM models and their procedures and I did, but what was a nice surprise was learning about the existence of the SMOTE method and imbalanced classification. This was a new area for me to study up on and I did not realize when I chose this Kaggle dataset that I would be dealing with an imbalanced classification issue. I felt that I tried to understand it to the best of my ability and am happy that I learned about its process.

To improve or change anything about this project I would spend more time learning about the SMOTE method and also perhaps look into using sci-kit learn instead of R as I ran into issues with reproducing my results.

## VII. References

[1] Kaggle. (2016). Diagnose Specific Language Impairment in Children [Online].
Available: https://www.kaggle.com/dgokeeffe/specific-language-impairment
[2] machinelearningmastery. (2020). SMOTE for imbalanced classification with Python [Online].
Available: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification
[3] machinelearningmastery. (2020). Tour of Evaluation Metrics for Imbalanced Classification [Online]
Available: https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification