**GROUP NUMBER:** 9

**GROUP:** David Elimelech, Nicholas Magal, Rafael Marques Braga

**CLASS:** ECE 548 - Machine Learning

**DATE:** November 2, 2018

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**TITLE:** Imbalanced Class Representation, Precision and Recall for Performance Evaluation

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ABSTRACT**

The goal of this experiment was to show the effects of a machine learning induced classifier on a training and testing set in which the class distribution is broken up in a 90% majority class and 10% minority class split (different from each data set). The classifier used was a 3-NN which we implemented from scratch in MatLab. The problem regarding this class imbalance was that the classifier was much more likely to label a testing set example as the majority class (bias). We attempted to reduce this issue by implementing a majority-class undersampling mechanism based on one-sided selection using Tomek links. Calculating the error rate of a classifier on such domains does little to help evaluate a classifier’s performance because a classifier that always by default labels an example as positive, where the positive class is the 97% majority, means the classifier would have an error rate of 3% but is useless. Thus, we used precision and recall metrics to compare the performance of a classifier before and after data pre-processing using Tomek link undersampling.

All attribute values were normalized between the range of 0-1 using the basic normalization equation of (x-min)/(max-min). Also, the training and testing set were stratified so that there was a 90/10 majority vs minority split in both the training and the testing set. (With exception to the self-created domain for representational purposes).

This was all evaluated on 3 different domains. One domain we created ourselves; the other three were from the UCI repository: the Abalone data set, the bank authentication data set, and the yeast data set.

**IMPLEMENTATION OF K-NN AND MAJORITY-CLASS UNDERSAMPLING**

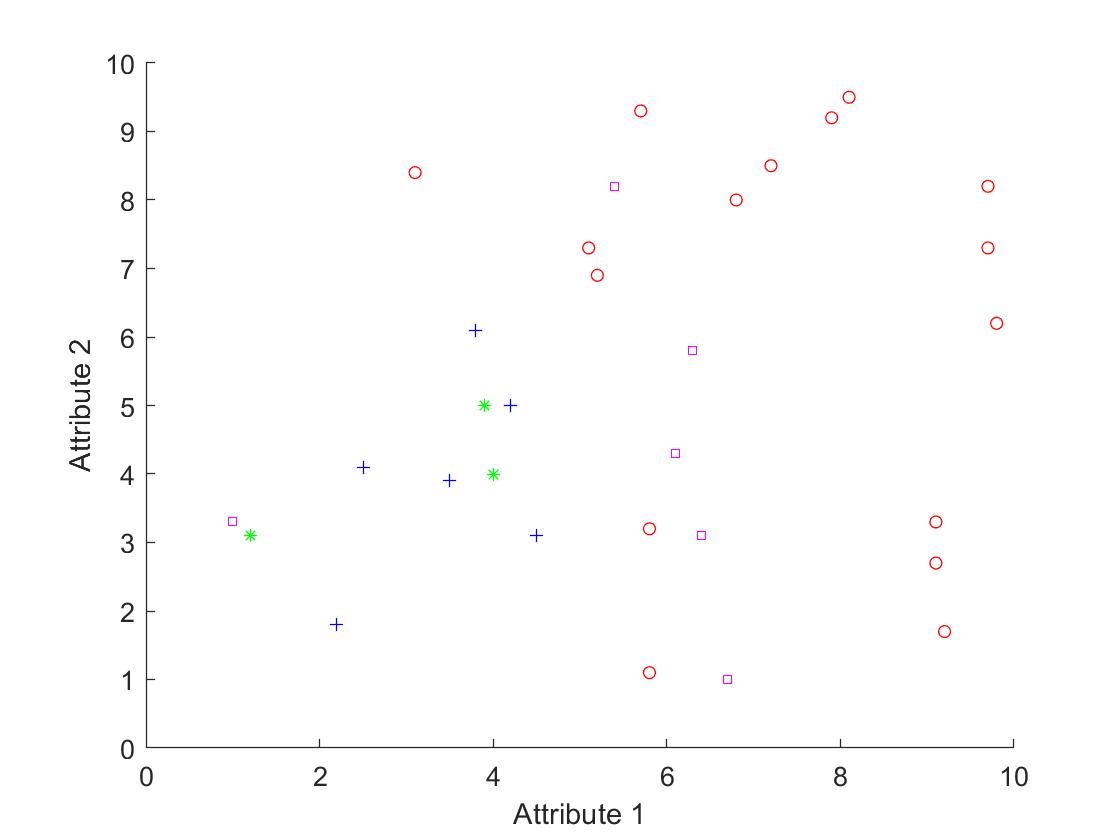
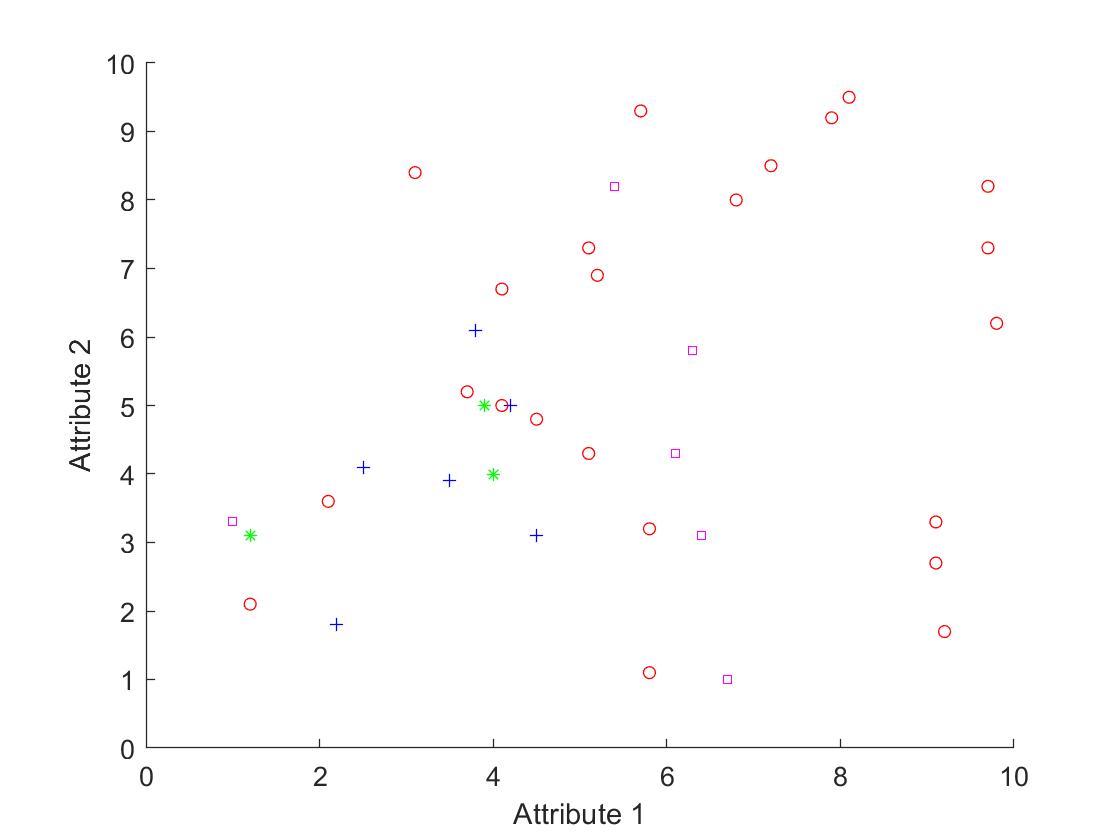
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

See attached code, implemented in MatLab from scratch.

**RESULTS AND ANALYSIS**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The self created domain contained 29 training domain examples and 11 testing examples. In the training domain 6 examples were positive (the minority class) and 23 examples were negative(the majority class). We found both precision and recall to increase after Tomek link undersampling. Precision went from NaN (not a number) to .6 and recall went from 0 to 1. Precision was initially not a number because the classifier was never classifying any training example as positive (either as a false positive or true positive). After Tomek link removal, as can be seen on the graph, some examples are now classified as positive. The results of this are somewhat not surprising for us, as we surmised after the removal of noisy data that the classifier would perform better than before preprocessing.



*Red examples are negative training examples, blue examples are positive training examples, green stars are positive testing examples, and pink squares are negative testing examples.*

The Abalone domain contained 4177 examples in the domain. We combined classes together to achieve a 90/10 split between majority and minority classes. The training and testing set were made forming an 80% training and 20% testing split. In the each, the examples were sampled in a stratified manner to uphold the 90/10 split between majority and minority. This pattern of stratification is repeated throughout all of the next domains.

Before undersampling was performed precision was measured at 0.275, recall was measured at 0.15068, sensitivity was measured at 0.15068, and specificity was measured at 0.96194 percent. The error rate of the classifier was 10% and accuracy was 89%.

After undersampling was performed precision was measured at 0.26, recall was measured at 0.23, sensitivity was also 0.23, specificity was 0.93, error rate was 12%, and accuracy was 87%.

A interesting point is that after the Tomek link undersampling the precision decreased at the expense of the recall. This makes sense, because if recall increases, the number of false negatives decreases while number of false positives increases. Which is mostly what you would like as a result in an imbalanced domain: for your classifier to be more complete, as false negatives are usually more costly than false positives in the case of the minority class.

The result on the Bank Authentication domain were extremely interesting. On the first run the precision and recall were both of value 1, and after Tomek link undersampling was performed the precision and recall were still 1. This means that the number of false positives and the number of the false negatives were both 0, and that the classifier was working perfectly before and after Tomek links were removed. This supports our hypothesis that tomek link undersampling doesn’t help performance unless the domain is noisy.

To test this idea we added artificial noise to the domain by changing the the class label on approximately 30% of the class. Precision and recall before Tomek link undersampling was .53 and recall was .77. As expected, after the tomek link undersampling mechanism was implemented (which removed 88 examples) the precision decreased to .47 and and the recall increased to .88.

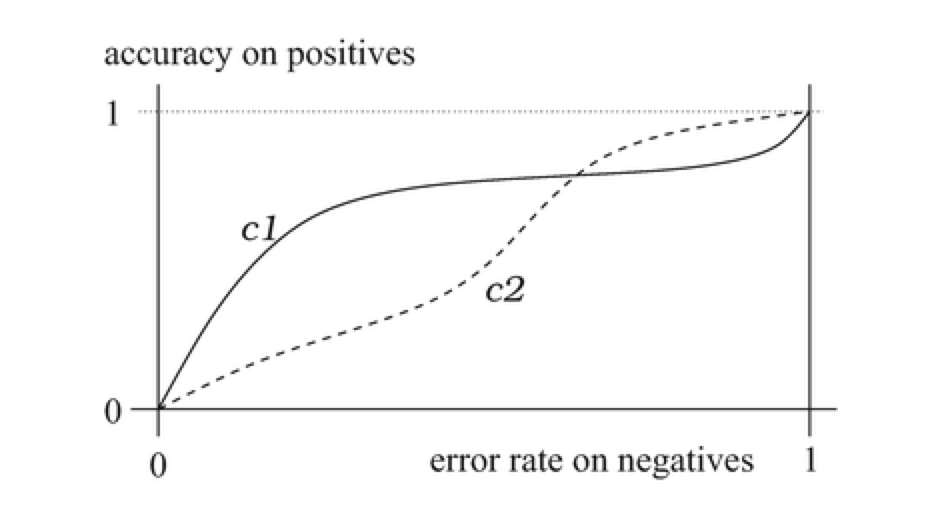
**CONCLUSION**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In conclusion, in domains where there is a large under-representation of a given class, it is likely to misclassify testing examples as the majority class due to the inevitable bias in learning. One possible way to mitigate this effect is to instate the majority-class undersampling mechanism of one-sided selection using Tomek links. This will reduce the disparity in the number of examples from the majority class versus minority class and thus aim to help create more well-ordered data. We found that it was important to stratify the training set as well as the testing set in order to have the two sets representative of the actual domain.

Another interesting thing we found out was the effect of increasing the recall on the precision. We found that these two were often connected: where increasing the recall would come at the expense of the precision, and vice versa. This can be also summarized in the graph below.

But as we saw on a few occasions, removing the Tomek links made little to no difference in the results of the classifier as when we did no preprocessing of the data at all. We believe this is because the Tomek link undersampling method mainly removes examples that are either noisy or lie along the class decision boundary, as those are the most common cases/areas for nearest neighbors to have opposing classes. Thus, in domains that are already well-ordered, Tomek link removal usually does not remove that many majority examples, and the classifier is already set to do nearly as well after pre-processing as it did before. But if anything, the mechanism will help provide support for the minority class, especially in the border region between the two classes. In contrast, in domains with more noise, Tomek link undersampling helps make a bigger difference between the performance of the classifier on the pre-processed vs post-processed data.



**FURTHER STUDIES**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

An interesting topic to consider would be what mechanism to use when recall matters more versus when precision matter more. Consider a classifier that classifies a life threatening situation, such as cancer. At this point the classifier can not tolerate false negatives very much as doing so places the patient’s life at considerable risk. In this case, the value of recall is very important and should be as close to 1 as possible, while the precision value is not as important. Conversely, consider a e-commerce website that utilizes a recommendation system. In this case, the number of false positives better be low because otherwise the customer may not pay attention to future examples, so the precision must be close to 1.

In connection to the importance of precision and recall, and the tug-of-war relationship they experience, is the concept of the Fβ criterion. Measuring the effectiveness of a classifier using the Fβ criterion would be a helpful metric to have because using the Fβ criterion allows us to select the importance of recall vs precision using β, which would be helpful when using different types of domains (E.g. Medical vs recommendation system).

