Radioxenon monitoring for verification of the Comprehensive nuclear-Test-Ban Treaty

Nitish Srivastava(Y7268) **Varunesh Mishra (Y7492)**
Department of Computer Science and Engineering, IIT Kanpur

April 20, 2010

Abstract

This project is about developing and testing data mining techniques to verify worldwide compliance of the global ban on nuclear tests. The dataset consists of Radioxenon measurements from five CTBTO monitoring sites. The key features of this dataset include imbalanced representation of the two classes. We experiment with several classifier types like J48 with adaboosting, single hidden layer neural networks, support vector machine and decision trees. We experiment with three methods for handling data imbalance: the Smote method for SVMs, changing the probabilty cutoff for classification and randomly upsampling and downsampling the classes.

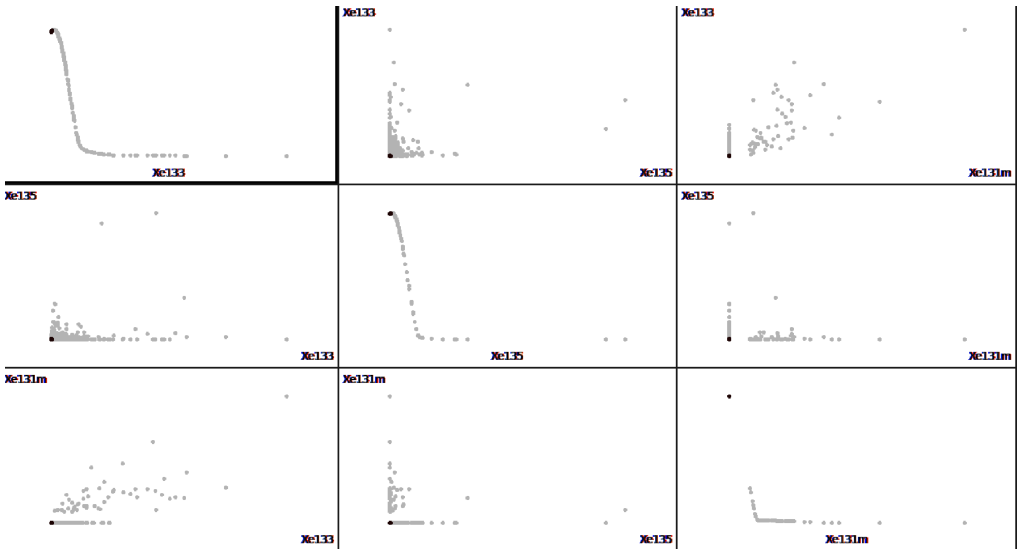
1 Introduction

For verification the compliance of the Comprehensive Nuclear-Test-Ban Treaty (CTBT), remote detection and measurement of radioactive forms of noble gas called radioxenon is employed. This gas is emitted from nuclear sources including nuclear explosion. The principle of such detection is that certain combinations of the four radioxenon can be finger prints of a nuclear explosion. Several remote sensing stations have already been deployed, whose purpose is to monitor the atmosphere for traces of Xenon isotopes indicative of nuclear explosions. However, the problem is complicated in two aspects. First, the detection station could be well over a thousand kilometres away from the explosion location, the gas emitted in a explosion could be remarkably degraded due to radioactive decay during weeks of atmospheric transport process, making the figerprints less likely to be detected. Second, there could be other radioactive sources emitting radioxenon, such as nuclear power plants, medical isotope production facilities, or various types of weapons. The radioxenon given by sources other than nuclear explosion is treated as background. The general task of the project is to devise methods to distin- guish between those radioxenon measurements that are due purely to normal environmental emissions or background (B) from those measurements that contain the signature of an explosion combined background (B+E).

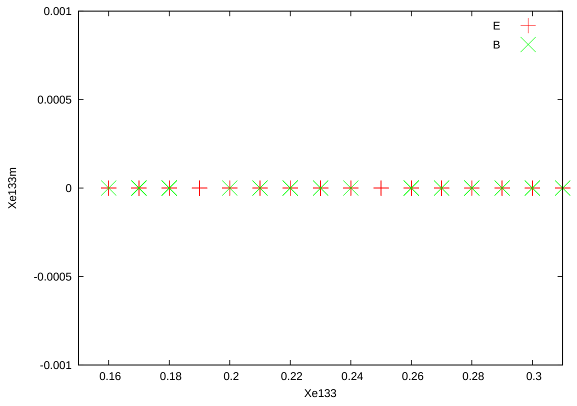
2 Properties of the dataset

The overall dataset consists of Radioxenon measurements of 4 radioactive Xenon isotopes at five observation stations. The training set consists of 8695 such vectors each labeled B (background) or B+E (background+explosion). However, there is a severe imbalance of data across the two classes (only 623 of these (7%) belong to B). The data information for each station separately is given in the table below

Station	B	B+E	% of B
v	115	1589	6.74
w	14	763	1.80
x	210	2090	9.13
y	40	1169	3.30
z	244	2461	9.02



(a) Scatter plot for all points for Station V. The small dark spots belong to the minority class. The majority is shadowed

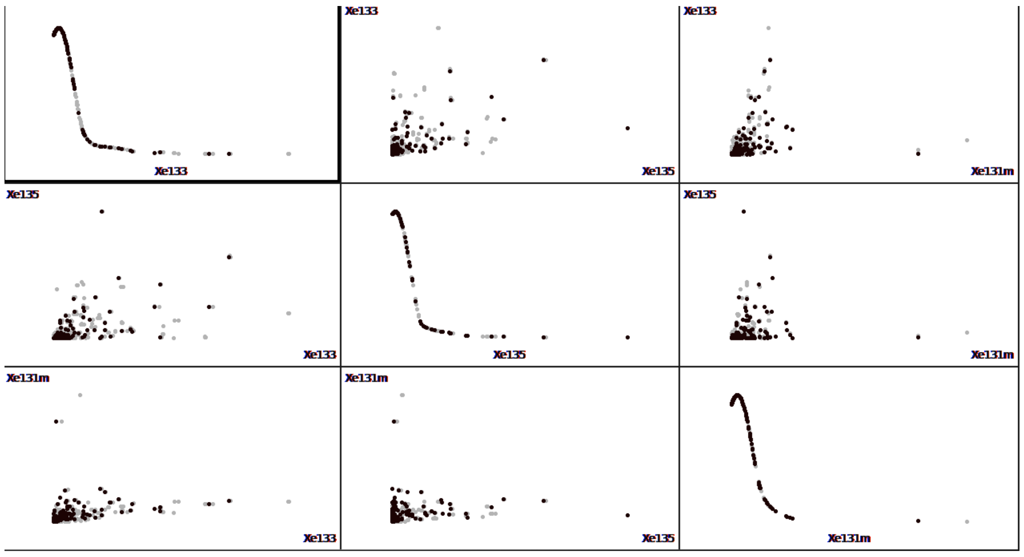


(b) Zoomed in view alonf 'Xe133' and 'Xe133m'

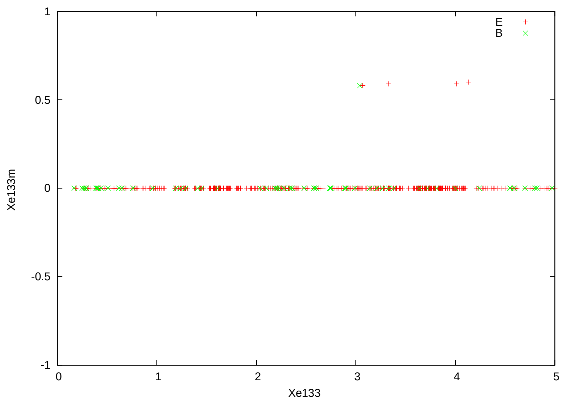
Figure 1: Visualizing the dataset: Scatter plot for Station V dataset

It becomes important to handle this disparity because any reasonable statistical classifier would have to work very hard to achieve an accuracy of more than 93% which will be obtained by the trivial classifier that classifies eveything into category (B+E).

Another salient feature of the dataset is that the points have a very high density in a small region of the feature space. For example Figure 1(a) shows the datapoints in the space of two observation variables for station V. Figure 1(b) shows a zoomed in view of the feature space, showing that both labels almost overlap at the same points. This visualization shows that the data is difficult to separate. Also the prior probability distribution of the classes at each station was found to be different. So we choose to train the models separately for each station. For example, the data distribution for station Z shown in Figure 2(a) is clearly quite different from the distribution for station V shown in Figure 1(a). For station Z both the classes are equally distributed in space and overlapping with each other therefore it is very difficult a design an efficient classifier for Z. Figure 2 shows the datapoint for station Z.



(a) Scatter plot for all points for Station *Z*. The dark spots belong to the minority class. The majority is shadowed to allow visibility

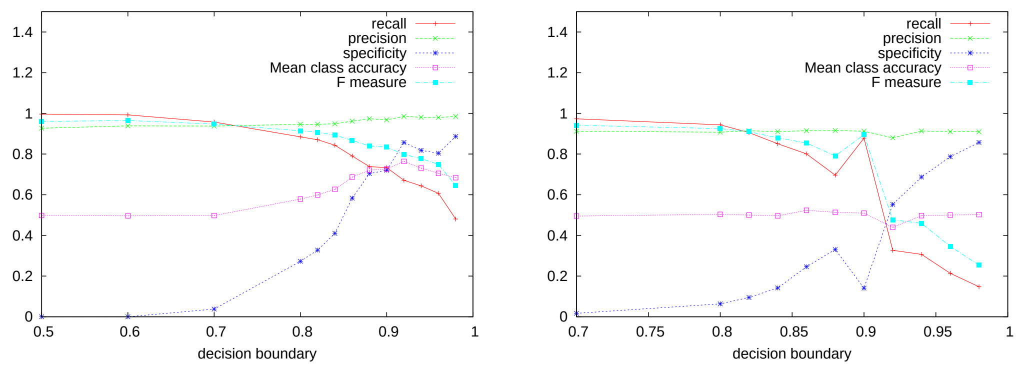


(b) Zoomed in view along ’Xe133’ and ’Xe133m’ showing clear overlap of labels

Figure 2: Visualizing the dataset: Scatter plot for station Z dataset

3 Handling Class Imbalance

There are many methods to preprocess the data for handling class imbalance. One is to electively under-sampled the majority class while keeping the original population of the minority class and another can be to over sample the minority class keeping fix the majority class. Mixture of the above methods had also been used. We preprocessed the data with Smote - Synthetic Minority Over-sampling TEchnique [2] for use in the SVM classifier. It is an over-sampling approach in which the minority class is over-sampled by creating synthetic examples rather than by over-sampling with replacement. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the *k* minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the *k* nearest neighbors are randomly chosen. We use the weka toolkit to preprocess our data using Smote method. The number of nearest neighbour was set to 3, we avoided using a large number so that it does not introduce the instances which overalps the majority class. And also using a small number like 1 for nearest neighbour will not be useful because in that case it will be just replicating the instances of minority class which will be not helpful to improve the performance of classifier. We sampled the data such that the



(a) Station *V* (b) Station *Z*

Figure 3: Performance measures for classification using neural network. The tradeoff between specificity (blue curve) and recall (red curve) is clearly visible. The optimal value can be taken as the point of intersection of these curves.

number of instances of both the classes are same. A simpler way to handle imbalance for soft classifiers is to change the decision boundary so as to favour the mionority class. We used this method for working with Neural networks, k-NN and decision tree classifier models. The Smote method was only used for SVMs which are hard classifiers.

4 Task 1

4.1 Neural networks

4.1.1 Choosing number of nodes in the hidden layer

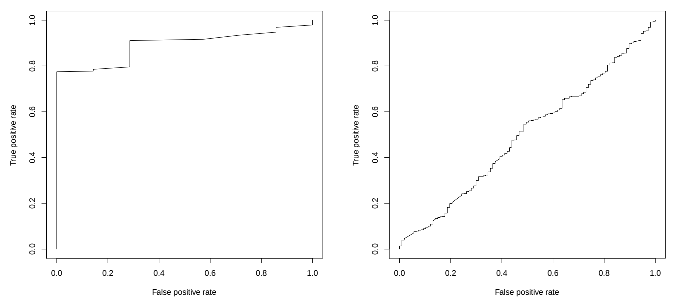
We use a neural network with 4 input nodes, a single hidden layer containing 5 nodes and one output node with logistic activation. The number of nodes in the hidden layer k was decided after experimenting with k and finding the effect on different error measures. The results showed that there was no major impact on the average case performance with respect to different error measures but a relatively high value gave more stable results. Hence the value of 5 was chosen.

4.1.2 Tuning the decison boundary

The output of the neural network gives the probability of the input belonging to class $B+E$. Due to imbalance in the data in favour of class $B+E$, these probability values never fall below 0.5 mark. Hence the decision boundary corresponding to Bayes rule is not the best for this case. It is to be noted that the Bayes rule provably gives the *best* classifier, where best refers to maximizing total accuracy of classification. However, in the context of this dataset, *best* refers to having a good mean class accuracy, i.e. maximizing the average accuracy over all classes where the accuracy for each class is normalized by the size of that class. This metric tries to counter the effect of imbalance in the data. Experiments were conducted with varying values of the decision boundary to decide the best boundary for each station. Figures 3 show the results for different stations and compares the performance of the classifier on metrics such as recall, precision, specificity, mean class accuracy and F-measure. A reasonable way to estimate the decison boundary is to locate the probablity value at which the recall and sensitivity values meet. This intuitively corresponds to the equivalent of a Bayesian equi-probable contour for this definition of the *best* classifier.

Station	V	W	X	Y	Z
Decision Boundary	0.88	0.97	0.89	0.95	0.91

These different values were then used to construct a classifier for the entire dataset.



(a) Station *W*   (b) Station *Z*

Figure 4: ROC plots for classification using neural network

4.1.3 Discussion of results

In order to facilitate the discussion of the results we show the Receiver Operator Characteristic (ROC) curves for each station's classifier (Figure 4). AUC measures the overall confidence in the classifier as it takes into account the behaviour at different decision boundaries. We use it as the metric to compare different classifiers. In all the experiments we randomly sample 50% of the data for use as training set and the remaining half is kept for testing. The table below shows the AUC (Area under the ROC Curve) values for each case

AUC at Station	V	W	X	Y	Z	**Overall**
Neural Net	0.7851	0.8906	0.6704	0.7675	0.4968	**0.6717**
Decision Tree	0.6712	0.6216	0.6004	0.6143	0.4334	**0.5661**
3-NN	0.6407	0.6122	0.5742	0.5458	0.4225	**0.5395**

Neural network not only gives the best overall performance but is better than other classifiers at each station. Given the kind of dataset that the we are working on, this can explained systematically.

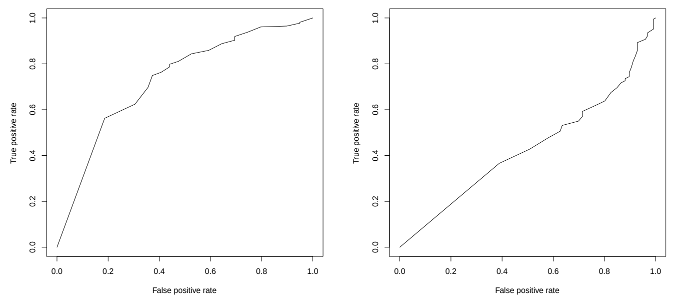
4.2 Comparison with other classifiers

4.2.1 Decision Trees

The results for the ROC curves is shown in Figure 5. The brancing factor and depth were controlled using an impurity gain threshold which was tuned to give an appropriately sized tree. The fact that decision trees do a hierarchical splitting can be used to understand why they do not perform well on this dataset. The activity of different isotopes, as recorded in this dataset, is not inherently hierarchical, i.e., there is no strong dependence on any one towards predicting an explosion. Neural networks, due to a more pervasive linked strucuture are able to infer the dependencies more closely than decision trees. Intuitively, decision trees restrict themselves to a tree structure while neural nets have the freedom to train a DAG. This accounts for the additional power of neural networks.

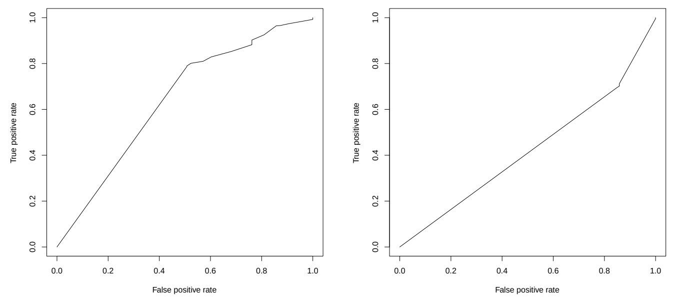
4.2.2 k-Nearest Neighbour

The ROC plots for 3-NN classifier are shown in Figure 6. The number of nearest neighbour was set to 3. We avoided using a large number so that it does not introduce the instances which overlaps the majority class. And also using a small number like 1 for nearest neighbour will not be useful because in that case the soft classification, on which the decision boundary shifting method relies, will not really be soft. The classification will be only into one class reducing k-NN to a hard classifier. Due to very low concentartion of the class "B", this classifier does not work as well for imbalanced data. The small number of nearest neighbors which we were forced to use made the classifier unstable producing large variations in accuracy over different random samples when cross-validating on the training set.



(a) Station *V* (b) Station *Z*

Figure 5: ROC plots for classification using decision trees



(a) Station *V* (b) Station *Z*

Figure 6: ROC plots for classification using 3-NN classifier

4.2.3 SVM

Support Vector learning is based on simple ideas which originated in statistical learning theory (Vapnik 1998). The simplicity comes from the fact that Support Vector Machines (SVMs) apply a simple linear method to the data but in a high-dimensional feature space non-linearly related to the input space. Moreover, even though we can think of SVMs as a linear algorithm in a high-dimensional space, in practice, it does not involve any computations in that highdimensional space. This simplicity combined with state of the art performance on many learning problems (classification, regression, and novelty detection) has contributed to the popularity of the SVM. SVMs have been used for imbalanced datasets in Akbani et al. [1].
Sigmoid kernel is quite a popular kernel for support vector machines due to its origin from nueral networks.

$$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + Coef0)$$

The sigmoid kernel becomes a PSD kernel only for some fixed range of parameters. Therefore it was very important for us to choose the parameters which make it a valid kernel. After studying about the kernel and using the results of Lin et al. [6] we decided to choose small γ and $Coef0$ value. Our assumption were futher justified by the tune function of the R which gives us a very similar values.

4.2.4 Parameter Optimization

We used inbuilt **tune** present in R [4]. It gives a parameter for the given model and test set. This generic function tunes hyperparameters of statistical methods using a grid search over supplied parameter ranges. The paramters returned for the sigmoid kernel were

$$\gamma = 0.1$$
$$coef0 = 0$$

4.2.5 Results

Using Station V

	Without Smote	Using Smote
Recall	0.9325117	1
Precision	1	0.02517306
Specificity	NaN	0.5268784
Mean Class Error	NaN	0.7634392
F measure	0.990909	0.04910988

Using Station W

	Without Smote	Using Smote
Recall	0.981982	1
Precision	1	0.3289646
Specificity	NaN	0.5675676
Mean Class Error	NaN	0.7837838
F measure	0.9650774	0.495069

Using Station X

	Without Smote	Using Smote
Recall	0.9086957	0.815195
Precision	1	0.3799043
Specificity	NaN	0.6217163
Mean Class Error	NaN	0.7184557
F measure	0.952164	0.5182768

Using Station Y

	Without Smote	Using Smote
Recall	0.9669148	1
Precision	1	0.421728
Specificity	NaN	0.6613226
Mean Class Error	NaN	0.8306613
F measure	0.9831791	0.5932611

Using Station Z

	Without Smote	Using Smote
Recall	0.9097967	0.6659267
Precision	1	0.9752133
Specificity	NaN	0.2077922
Mean Class Error	NaN	0.4368595
F measure	0.9527681	0.7914262

Support vector machines did not yield good performance. Without preprocessing the data, it mostly classified all the instances belonging to "B+E" class. Although it gives good accuracy but because of class imbalance in the data sets we cannot rely on this measure. With the other measures its performance is still not very satisfactory and consistent.

Even after using the Smote method for over-sampilng the data, the results obtained are not as good as other classifiers. It is mainly because most of the points of class B are overlapping with class B+E points, so it is very difficult to have good separating planes for these classes.

5 Task 2

For task 2, we have to build an optimal classifier for each station separately. In this task to handle class imbalance we upsample and downsample the classes by randomly choosing points from the data set. We tested with Adaboost and probability estimation using SVM.

5.1 Adaptive Boosting

AdaBoost - Adaptive Boosting, is a machine learning algorithm can be used in conjunction with many other learning algorithms to improve their performance. AdaBoost is adaptive in the sense that subsequent classifiers built are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. AdaBoost calls a classifier repeatedly in a series of rounds . Adaboost is an algorithm for constructing strong classifier as a linear combination of weak classifier.

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

where $h_t(x)$ is a weak classifier and α_t is weight

We used J48 [3] as weak classifier for adaboosting. J48 builds decision trees from a set of labeled training data using the concept of information entropy. It uses the fact that each attribute of the data can be used to make a decision by splitting the data into smaller subsets. J48 examines the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. To make the decision, the attribute with the highest normalized information gain is used. Then the algorithm recurs on the smaller subsets. The splitting procedure stops if all instances in a subset belong to the same class. We randomly upsample and downsample minority and majority class respectively to handle the class imbalance.

5.1.1 Discussion of Results

During up-sampling or down-sampling it is possible that instances of one class becomes much more than the other class. To get equal number of instances of both the classes we tuned the training sets for a optimum up-sampling and down-sampling parameter. Up-sampling parameter -*up* will repeat the number of instances of a class such that total number of instances becomes *up* times. Down-sampling-*down* parameter will down-sample the instances of a class such that total number of instances becomes *down* times the initial number of instances. Up sampling and down sampling parameters for each station are given in the table below.

Station	V	W	X	Y	Z
Up-sampling	3	5	3	4	2
Down-sampling	0.3	0.3	0.3	0.2	0.3

Area under ROC(AUC) obtained for each station separately is given in the table below. Results of SMOTE are better than random sampling because random sampling is just repeating points from the data set whereas SMOTE introduces new points on the basis of nearest neighbour approach. Smote generates synthetic instances in less application specific manner by operating in "feature space" rather than data space. So one can more rely on this approach to handle class imbalance.

Station	V	W	X	Y	Z
AUC with random Sampling	0.786	0.793	0.677	0.710	0.530
AUC with Smote	0.791	0.833	0.711	0.701	0.571

We might expect AdaBoost, with decision trees as weak classifiers, to work better on this dataset compared to other classifiers which rely on spatial locality of the data points (in the given or transformed space). This is beacuse the dataset is such that there is no clustering behaviour of points in the same class, i.e., data points of both classes are present quite close to each other in almost all regions of the space. This was observed by visualizing the data using visualization tools (¡put figure¿). Template matching methods (such as SVMs) separate classes by partioning the space of the datapoints (or a transformed space) and classifying new points according to this partioning alone. However, in case of AdaBoost, the weak classifiers partition the data space but their votes are then pooled and weighted. This effectively allows AdaBoost to develop classifiers for different regions of the space, quite independent of each other. AdaBoost takes subsamples of the data and constructs C4.5-decision trees for each subsample. Each decision tree is a weak learner and is assigned a weight. When classifying a new point, the algorithm collects decisons from each tree and decides accordingly. This method works particularly well on the dataset from station Z, where the classes are mixed together over a large region of the space. Neural networks too give good performance for a similar reason. The intermediate nodes could be effectively computing the classificiation decision of differe nt regions of the space. They however do not work as well on Z- data since the mix up of both classes is too high. Smote improves the performance in Adaboosting whereas in SVM degrades the performance because in the case of SVM it is possible that the new points introduced are overlapping with points of other class making it difficult for them to separate using SVM.

5.2 Probability Estimation using SVM

Because of huge class imbalance, hard classification will not be very helpful so it is important to get a likelihood of an instance belonging to a certain class and then varying the cutoff for classification to get a better classification. For generating probabilities we replaced class ”B” with 0 and class ”B+E” with 1. The output prediction is the probability that an instance belongs to a certain class.

5.2.1 Discussion of Results

For sampling previous parameters are used. Area under ROC(AUC) obtained for each station separately is given in the table below.

Station	V	W	X	Y	Z
AUC	0.560	0.583	0.512	0.517	0.412

The performance of SVM with this dataset is not good as there are lots of instances overlapping with each other. For better classification a better sampling methods is required to handle class imbalance.

6 Conclusions

We conclude that Neural networks and AdaBoost with decision trees as weak learners works best for this dataset. The probable explanations for the success of these methods and the failure of others are discussed in the report.

Station	Sampling Method	Classifier	AUC
V	Smote	AdaBoost	0.7910
W	Random	Neural network	0.8906
X	Smote	AdaBoost	0.7110
Y	Random	Neural network	0.7675
Z	Smote	AdaBoost	0.5710

7 Tools used

In this project we used the R statistical package [4] and associated open-source libraries for the implementations for neural networks, k-NN algorithms, SVMs and Decision trees. The Weka toolkit [7] was used for its implemetation of Smote. Ggobi software [5] was used for data visualization.

References

[1] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *ECML*, pages 39–50, 2004.

[2] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 2002.

[3] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[4] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. ISBN 3-900051-07-0.

[5] Deborah F. Swayne, Duncan Temple Lang, Andreas Buja, and Dianne Cook. Ggobi: evolving from xgobi into an extensible framework for interactive data visualization. *Comput. Stat. Data Anal.*, 43(4):423–444, 2003.

[6] Hsuan tien Lin and Chih-Jen Lin. A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. Technical report, 2003.

[7] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 1st edition, October 1999.