Large Scale Empirical Comparison of Linear Classifiers for Multi-class Problems

Ayon Sen Ashwin Karthi Narayanaswamy Anubhavnidhi Abhashkumar

***Abstract*—Linear classifiers, even though very simple, are popular for classification tasks. By nature they can only differentiate between two classes. But it is possible to extend their usage into the domain of multi-class problems. These classifiers are known to perfrom very well for many practical scenarios (both binary and multi-class). In this paper, we examine and study the performance of linear classifiers across different data sets for multi-class classification. Specifically, we compare the performance of Logistic Regression, Naive Bayes, Linear SVM, Weighted Majority and ECOC (Error Correcting Output Codes) constructed using Naive Bayes across various data sets. In addition to that, we show how these classifiers perform when the data sets have only numeric attributes, only nominal attributes and mixed attributes. We also study about the performance of these linear classifiers on large data sets and class imbalance problems. Our study shows that different linear classifiers perform better in different situations.**

I. Introduction

Linear classifiers predict the class value based on the linear combination of the features of an instance. If $\vec{x}$ is a feature vector, then the linear classifier predicts the class value as $\vec{y} = f(\vec{w}, \vec{x})$ where $\vec{w}$ is a weight vector that is learned by the classifier when it is trained. Linear classifiers have an advantage over other classifiers in terms of the time taken to classify as it calculates the output as a linear combination of input features. For a binary classification task with 2 dimensional feature space, a linear classifier tries to find a line that separates the two classes in the feature space.

Linear classifiers like Naive Bayes, Logistic Regression, Linear SVM and Weighted Majority are used in a plethora of classification tasks. In this study, we have tried to answer the following questions. If we have a data set in which the type of features are well known, which of these linear classifiers would provide the best accuracy? How do such classifiers perform for data sets with a large number of instances? If we use linear classifiers and scale it to multi-class classification, how does the model perform?

There are mainly two family of approaches to handle multi-class problems using linear classifiers like *one-vs-one* (OVO) and *one-vs-all*(OVA) [1]. The OVO approach divides a multi-class problem into as many binary problems as possible. It builds ${}^{m}\text{C}_2$ base classifiers for solving the problem having m classes. Each classifier in OVO discriminates between a pair of classes. In contrast, the OVA approach builds one classifier for each class, the so called *target class* and thus only requires m classifiers. Here the classifier discriminates the target class from the *default class* i.e., all the other $(m-1)$ classes. For our study we use OVA since it is less resource hungry and some of our data sets are very large. We analyze the accuracy of the classifiers over 25 different data sets. Cohen's Kappa and GMean are taken as the metrics of performance.

The rest of the paper is organized as follows. Section II highlights the different research work that has been done on the classifiers that we use for the study along with a brief discussion about OVO and OVA approaches. In Section III, we discuss how we have implemented the classifiers for our study. Section IV presents our findings while Section V discusses upon those findings. Then we briefly conclude in Section VI.

II. Related Work

This section presents the previous research works associated with the linear classifiers that we have used for this study. Since we use these classifiers in the multi-class problem domain, we discuss the research works that show how linear classifiers can be used for such domains.

Naive Bayes is a simple probabilistic classifier that applies Bayes' theorem with strong independence assumption between the features. Even though the independence assumption made by this classifier can be incorrect, in practice it performs really well compared with other classifiers. It has been modified over the years to suit particular application domains. Lewis [2] provided a modified version of this classifier which is specially useful for text classification. He also showed how integer-valued features can be handled in such a setting. In [3], the author showed that Naive Bayes does not always scale well with larger databases. He then proposed a decision tree hybrid called *NBTRee* to improve performance for such cases. The authors of [4] also do an empirical study of such classifiers. They found that the accuracy of such classifiers is not directly correlated with the degree of feature dependencies.

Logistic regression is a probabilistic statistical classification model. It uses a discriminative approach and generally takes lesser number of training instances to converge than Naive Bayes [5]. The authors of [6] also showed that for binary classification, boosting can also be explained in terms of logistic regression.

Support Vector Machines [7] are based on the concept of decision planes that define decision boundaries. In a linear SVM, the decision planes define a boundary that separates two set of classes. The distance from the decision surface to the closest data point defines the margin of the classifier. The decision boundary is selected such that this margin is maximized. A single Support Vector Machine(SVM) can represent only a binary classification task. Multi-class SVMs are usually implemented by combining several two-class SVMs. Some of the famous methods as mentioned in [8] are OVO, OVA, Directed Acyclic Graph (DAG), and Error Corrected Output Coding (ECOC). For this study, we have used the OVA approach.

The weighted majority algorithm predicts the output by evaluating the predictions of a pool of experts. This paper [9] shows how weighted majority can be used for linear classification. It initially assigns a positive weight to each expert in the pool of algorithms. The algorithm forms its prediction by comparing the total weight q_0 of the experts in the pool that predict the class 0 to the total weight q_1 of the algorithms predicting 1. When the WM makes a mistake, the weights of those experts of the pool that disagreed with the label are each multiplied with a fixed 'β' such that $0 \leq \beta < 1$.

Error Correcting Output Codes (ECOC) is an ensemble approach where the class representation is changed in order to get better results. It was first proposed by the authors of [10]. They showed how classes represented by multi-bit code words can help to improve the performance. They also came up with code words for different number of classes. In [11], the authors present a heuristic method for learning the code words based on a hierarchical partition of the class space that maximizes a discriminative criterion. They make a trade off between optimal code word separation with maximum class discrimination in partitions.

OVO and OVA are the two main approaches that are used to solve multi-class problems using linear (binary) classifiers. OVO has the advantage of handling easier decision boundaries since it only deals with data items of two particular classes. But this comes at the cost of added resources (more classifiers required in comparison to OVA). When predicting the output classes both OVO and OVA has to combine the outputs of the classifiers that they use. OVO has another disadvantage here. There are multiple aggregations strategies for OVO like voting [12], weighted voting [13] and others [14], [15], [16]. Different strategies perform better in different situations. So, there is no clear winner. OVA, on the other hand, has mainly one aggregation strategy. Here, each classifier gives as output a probability of the data item belonging to the class that it was discriminating. The class with the highest probability is declared as the output class.

III. Methodology

In this section, we show how we have implemented the classifiers. There are two approaches of using binary classifiers to solve multi-class problems namely OVA and OVO. OVA uses less resources while OVO has the advantage of dealing with simpler decision boundary at the cost of using more resources. For this study we performed OVA classifiers with our linear classifiers. The implementations are described as follows.

A. Naive Bayes

As mentioned in Section II, Naive Bayes is one of the most popular linear classifiers. It has been used in a large number of real life problems. We used Wakaito Environment of Knowledge Analysis (WEKA) [17] to implement Naive Bayes. Unlike other implementations of Naive Bayes, in WEKA it can be directly used as both a binary or multi-class classifier. They use a OVA approach for this. To handle a numeric attribute, this class tries to estimate the distribution of that attribute [18]. From now on we will mention this classifier as NB.

B. Logistic Regression

For our experimental study of logistic regression, we again use Weka. Weka has multiple versions of logistic regression including SimpleLogistic [19], Logistic [20] and others. For our implementations we have used SimpleLogistic. It does not use a regularization parameter and is linear in nature. Some of the other logistic regression classifiers in Weka either use a ridge regularization parameter or is an extension of the linear version (multi-class classifier). Just like Naive Bayes in Weka the SimpleLogistic can handle multi-class classification automatically by using the OVA approach. Also, we changed the number of iterations for SimpleLogistic and implemented two versions. From now on we will call them LR1 and LR2. The parameters are shown in Table I.

Table I: Parameters for Logistic Regression

Name of Classifier	Iterations
LR1	100
LR2	500

C. SVM

For our experimental study of SVM, we again use Weka. For the purpose of multi-class classification weka have used the OVA version of SVM. In this project we have used 4 SVMs with different C values as mentioned in Table II.

Table II: Value of parameter C for various SVMs

Name of Classifier	C
SVM1	1
SVM2	10
SVM3	100
SVM4	1000

D. Weighted Majority

The weighted majority we implemented extends the idea [9] to multiple classes. The total weight q_i predicted for class i by the experts in the pool are compared and the prediction that has the maximum weight is taken as the prediction of the Weighted Majority Algorithm. The system under discussion uses the attributes of the dataset as the pool of experts. The experts are trained like a single level tree. Each attribute predicts all the classes for different nominal values and for different ranges in case of numeric type. Initially all the predictors are given equal weight of value 1. In the course of training, the weights of the predictors which predicted the wrong values are penalized by a factor of β which is given a value 0.95 . The value of β is a little high as the number of mistakes increases with the size of the training set and the weights of the predictors would decrease faster in case of large data sets.

E. ECOC

It was mentioned in Section II that ECOC changes the representation of the classes. It uses an ensemble of classifiers to decide individual bits of the code word for the output class. Since each bit can be either a '0' or '1', a linear classifier can be used. For our experimental studies, we chose Naive Bayes for this purpose. ECOC also does not perform very well when the number of classes is small. Usually the number of classes have to be greater than 11. Among the data sets that we chose for the study, three (arrhythmia, audiology and letter) fulfill that criteria. So, these were the only data sets that we tested with ECOC. One of the most important aspect of using ECOC is to have good code words. Such code words must have high Hamming Distance separation between them. For our study we chose the code words found in [21]. The code words could also have different lengths (15, 31 or 63) for the same number of classes. We used all three length code words for each data set except for arrhythmia. That data set only had 12 classes for which code words of only length 15 was provided. The implementation was made in Java programming language.

IV. Empirical Evaluation

In this section, we show our experimental results. Also, we discuss the data sets that were used along with the performance measures.

A. The Data Set

We select 25 data sets for simulation. Most of them belong to the UCI machine learning repository [22]. These data sets represent 25 different multi-class problems and can be obtained from http://sci2s.ugr.es/ovo-ova and http://repository.seasr.org/Datasets/UCI/arff/. We present brief characteristics of the data sets in Table III. The table shows the number of examples, number of attributes, the number of numerical and nominal attributes and the number of classes. As can be seen from the table, some of the data sets had a large number of instances (e.g. kdd_ipums_la_97-small, waveform-500). At the same time some data sets only had numeric attributes while other only nominal attributes. Others had a mix of both. Thus we can test the effectiveness of linear classifiers on data sets with different characteristics. This will help us gain insightful knowledge about how well these classifiers actually perform. Also, for our experiments we used stratified 10-fold cross validation.

Table III: Summary description of data sets.

Data set	#Example	#Attributes	#Numeric	#Nominal	#Classes
arrhythmia	452	279	197	82	12
audiology	226	69	0	69	24
autos	159	25	15	10	6
car	1728	6	6	0	4
cleveland	297	13	5	8	5
dermatology	366	33	1	32	6
ecoli	336	7	7	0	8
flare	1389	10	0	10	6
glass	214	9	9	0	6
kdd_ipums_la_97-small	7019	60	0	60	9
kdd_ipums_la_98-small	7485	60	0	60	10
kdd_ipums_la_99-small	8844	60	0	60	9
led7digit	500	7	0	7	10
letter	20000	16	16	0	26
lymphography	148	18	3	15	4
nursery	1296	8	0	8	5
pageblocks	548	10	10	0	5
penbased	1099	16	16	0	10
satimage	643	36	36	0	7
segment	2310	19	19	0	7
shuttle	2175	9	9	0	7
vehicle	846	18	18	0	4
vowel	990	13	13	0	11
waveform-5000	5000	40	40	0	3
zoo	101	16	0	16	7

B. Performance Metrics

Several measures exist in the literature for evaluating the performance of a classification framework. In this work, we use classification accuracy or simple accuracy, Cohen's Kappa and G-mean as performance measures.

- **Accuracy** also called the classification rate, is the number of correctly classified instances compared to the total number of instances. It is the most commonly used metric for assessing performance of classifiers.
- **Cohen's Kappa** is an alternative measure to classification rate. It compensates for random correct classifications. Cohen's Kappa evaluates the portion of hits that can be attributed to the classifier itself relative to all the classifications that cannot be attributed to chance alone. It can be calculated by making use of the resulting confusion matrix (Table IV) in a classification task.
  Cohen's Kappa is calculated as follows:
  $$\text{Kappa} = \frac{n\sum_{i=1}^{m} h_{ii} - \sum_{i=1}^{m} T_{ri}T_{ci}}{n^2 - \sum_{i=1}^{m} T_{ri}T_{ci}} \qquad (1)$$
  where h_{ii} is the number of true positives for each class, n is the total number of examples, m is the number of class labels and T_{ri} and T_{ci} are the row's count and column's count respectively. Cohen's Kappa ranges from -1 through 0 to 1. These values indicate total disagreement, random classification and perfect agreement respectively.
- **G-mean** is a measure of classification performance where each class is equally represented in the evaluation measure. It has been known to be an effective measure to evaluate the impact of imbalanced data. G-mean can be defined as follows for multiple classes:
  $$\text{G-mean} = \left(\prod_{i=1}^{m} \frac{h_{ii}}{\sum\limits_{j=1}^{m} h_{ji}} \right)^{1/m} \qquad (2)$$

Table IV: Confusion Matrix.

	C_1	C_2	$\ldots$	C_m	Total
C_1	h_{11}	h_{12}	$\ldots$	h_{1m}	T_{r1}
C_2	h_{21}	h_{22}	$\ldots$	h_{2m}	T_{r2}
$\vdots$			$\ddots$		$\vdots$
C_m	h_{m1}	h_{m2}	$\ldots$	h_{mm}	T_{rm}
Total	T_{c1}	T_{c2}	$\ldots$	T_{cm}	T

C. Results

Here, we present our experimental results. Table V shows the average results for each of the classifiers. Each cell in the table represents the average value of that performance measure over all the data sets for each of the 10 folds. The best results are shown in bold. This shows that LR2 has the highest accuracy and Cohen's Kappa value while SVM2 has the highest G-mean value. Also, the accuracy and Cohen's Kappa values for LR1, LR2, SVM1, SVM2 and SVM3 are pretty close to each other. For G-mean values, LR2, SVM1, SVM2, SVM3 and SVM4 are closer. In both cases, the other classifiers perform significantly worse.

Table V: Comparison between different linear classifiers for multi-class classification. The best results are shown in bold.

Classifier	Average Accuracy (%)	Average Cohen's Kappa (%)	Average G-mean (%)
NB	75.42	59.85	68.96
LR1	78.14	60.27	68.19
LR2	**80.44**	**63.69**	73.53
SVM1	77.93	62.27	73.46
SVM2	78.05	63.33	**74.31**
SVM3	75.80	61.51	73.86
SVM4	73.96	58.66	73.25
WM	53.96	26.35	80.58

Since we tested ECOC only on three data sets, those results are shown separately (Table VI). The table shows the average accuracy for each of the data sets over the 10 folds. Also, we show the length of the code words used beside the name of the data set. For both audiology and letter this length varies between 15, 31 and 63. But for arrhythmia code word length was fixed at 15. The reason behind this is mentioned in Section III.

Table VI: Results for ECOC. The length of the code words used is shown in parenthesis after the name of the data set.

Data Set	Accuracy (%)	Cohen's Kappa (%)	G-mean (%)
arrhythmia (15)	73.34	51.12	73.21
audiology (15)	61.63	53.72	88.87
audiology (31)	76.78	70.18	88.58
audiology (63)	76.78	70.35	88.31
letter (15)	56.92	53.96	53.73
letter (31)	63.11	60.38	61.82
letter (63)	63.22	60.94	63.66

V. Discussion

This section discusses our experimental findings. At first we look at the overall results. Then we look at data sets with only nominal attributes, only numerical attributes, and both nominal and numerical attributes. Finally we discuss the results gained from ECOC.

A. Overall Comparison

As mentioned earlier, we used two different sets of parameters for Logistic Regression. The comparison between these two versions for all the data sets are shown in Figure 1. It can be seen from the figure that for most of the data sets, LR1 has higher error rate that LR2. Similarly, for most of them LR1 has a higher G-mean than LR2. This shows that the performance of LR2 is much better in terms of both error rate and G-mean. It is well known that logistic regression performs better than Naive Bayes if we have limited amount of training instances. Since, a lot of our data sets had limited number of instances, it is no surprise that logistic regression performs better than Naive Bayes.

Like mentioned earlier for the SVM part we used 4 SVMs with different values of C. We found out that the first two SVMs, SVM1(C=1) and SVM2 (C=10) gave the best results. The comparison of error rate and G-mean of these two classifiers are shown in Figure 2. From the graph it can be seen that the difference in performance in terms of error rate is more compared to difference in performance in terms of G-mean. This is most likely due to the fact that models with higher value of C are overfitting. In the subsequent section we will see the reason behind why SVM performs worse than Logistic Regression.

The error rate of weighted majority was greater than other classifiers and also the performance was lower than expected. This is because most implementations of weighted majority use different classifiers as the predictors. We used only the attributes as the predictors as the study focused on linear classifiers. It turns out that attributes are poor predictors individually. Most of the attributes could not predict with high accuracy as this implementation of weighted majority could never take into account the dependency between attributes. Class imbalance affected the performance of weighted majority. For a given attribute value, the weight associated with the class that has maximum number of instances is penalized less and the weight of the class that has the minimum number of instances is penalized more. These reasons contribute to the high error rate of Weighted Majority.

B. Comparison for Nominal Only Attributes

Now we discuss our findings for data sets with only nominal attributes. There were eight such data sets. The average accuracy and average G-mean for all the classifiers over these data sets is shown in Figure 3. It can be seen from the figure that LR2 still outperforms all other classifier in terms of accuracy. But for G-mean NB has the best result. So, we can conclude that when all the attributes are nominal Naive Bayes usually performs better for data sets which have class imbalance problem but higher accuracy can be gained by using logistic regression. Another interesting observation was that only in the case of nominal attribute SVM1 performed better than all the other SVMs in terms of accuracy.

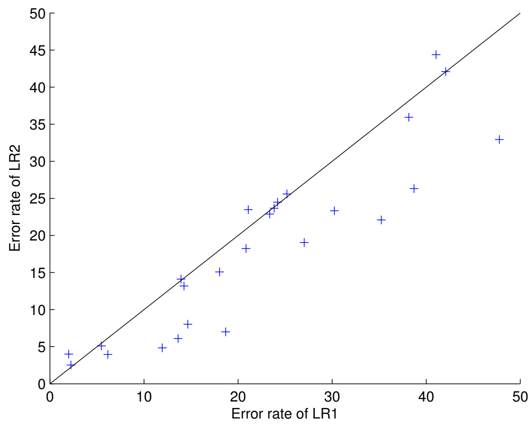
The accuracy of Weighted Majority was comparable with the other classifiers when the data sets did not have class imbalance and all the values were defined. The error rate was high for kdd data sets as these data sets have missing class and attribute values. Because of this, those instances in which an attribute value was missing could not be considered for training and those values which had class values missing could not be used. The error rate for the flare data set was high because the number of class imbalance.

From the Figure 3 we see that SVMs perform worse than LR2 and this can be attributed to the fact that the nominal attributes had to be converted into real valued features. This conversion might not have been done appropriately by Weka and this may cause degradation in performance of SVMs.

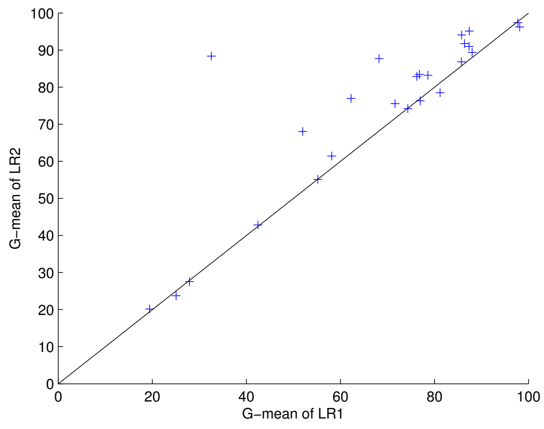
C. Comparison for Numeric Only Attributes

We present our findings for data sets with only numeric attributes here. The results are shown in Figure 4. For these data sets most of the classifiers except WM perform similarly. But overall SVM2 has the highest accuracy and G-mean suggesting that for numeric attributes SVM performs the best.

Contrary to the previous Figure 3, here SVM does perform better than both LR1 and LR2. This can again be shown as

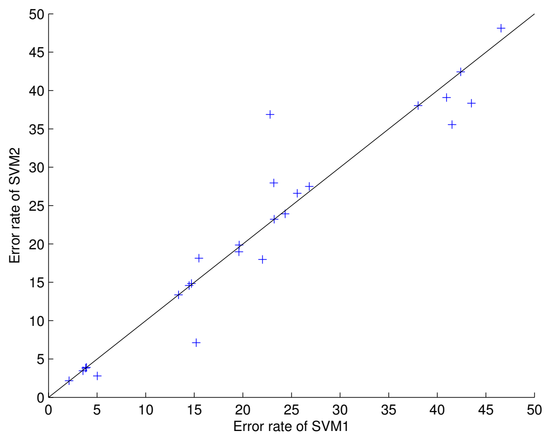


(a) Error Rate

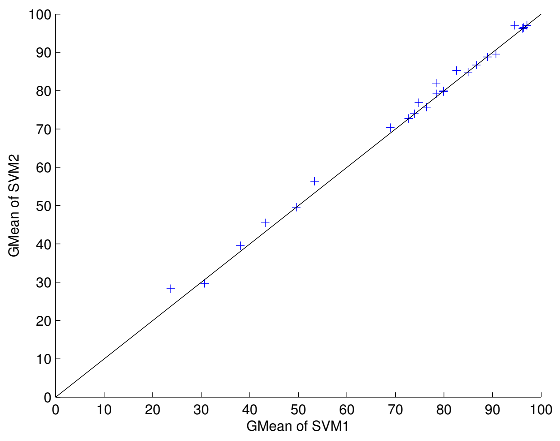


(b) G-mean

Figure 1: Comparison of error rate and G-mean for all data sets for LR1 and LR2.



(a) Error Rate



(b) G-mean

Figure 2: Comparison of error rate and G-mean for all data sets for SVM1 and SVM2.

proof that conversion from nominal to real value caused the performance to be poorer for SVMs on data sets with nominal only attributes.

D. *Comparison for Both Nominal and Numeric Attributes*

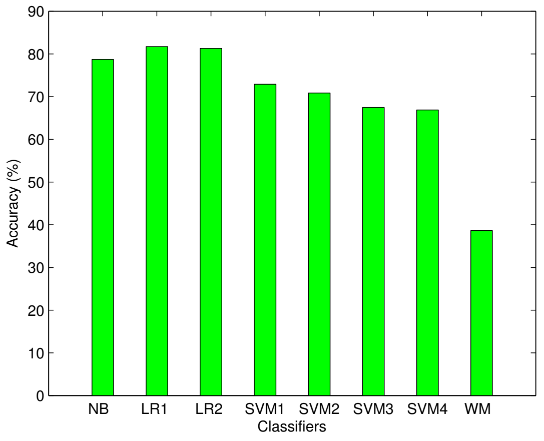
Now we look at data sets with both nominal and numerical attributes. The results shown in Figure 5 show that again Logistic Regression performs the best. But this time LR1 has higher accuracy while LR2 has better G-mean value. As mentioned in previous two subsections the conversion from nominal to real value caused degradation in performance for SVMs.

E. *Comparison for Large Data Sets*

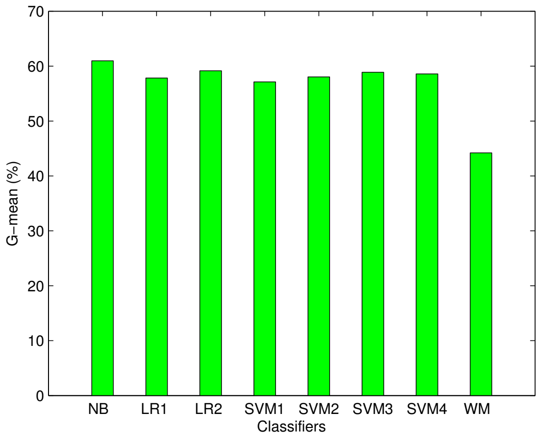
Finally we look at large data sets. The results are shown in Figure 6. An interesting observation over here is how different both the graphs are. In terms of accuracy the classifiers which did well are NB, LR1 and LR2 but in terms of G-mean we have SVM2, SVM3 and SVM4 as the better classifiers along with NB. This suggests that these data sets suffered from class imbalance. While the SVM classifiers were better at identifying classes that had less instances, it failed to get better accuracy for the classes which had more instances. This led to the mismatch in the figures.

F. *ECOC*

Now we discuss our findings regarding ECOC. While testing ECOC, we used Naive Bayes as our baseline classifier. From the results presented from Table VI, it can be seen that with the increase of code word length the performance of ECOC improves. But from code word length 31 to 63, the improvement is not that significant. This shows that for a particular number of classes the gain from using higher length code words after a specific length is not that helpful.

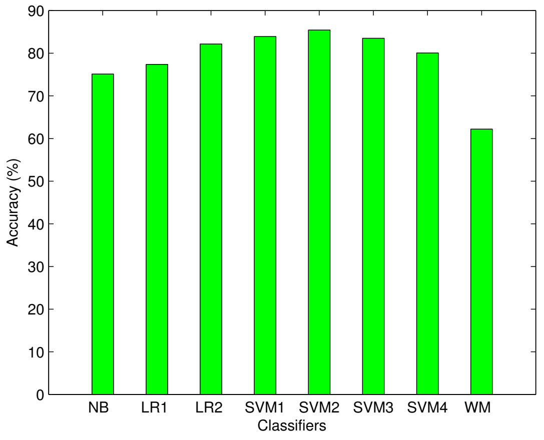


(a) Average Accuracy

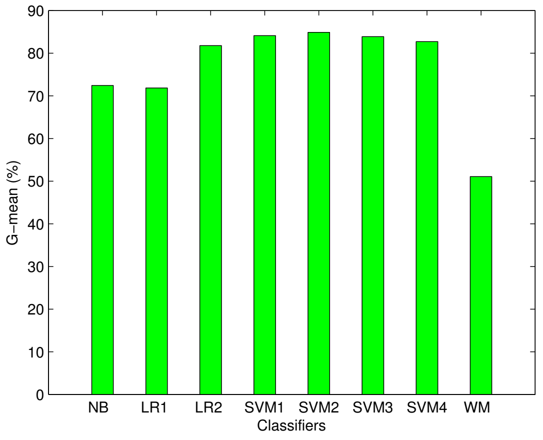


(b) Average G-mean

Figure 3: Results for data sets with only nominal attributes.



(a) Average Accuracy



(b) Average G-mean

Figure 4: Results for data sets with only numerical attributes.

We also wanted to see if ECOC performed better than Naive Bayes. We found that ECOC performed better than NB for two of the three data sets. Since, the number of data sets for comparison was small, no significant conclusion could be drawn from this comparison.

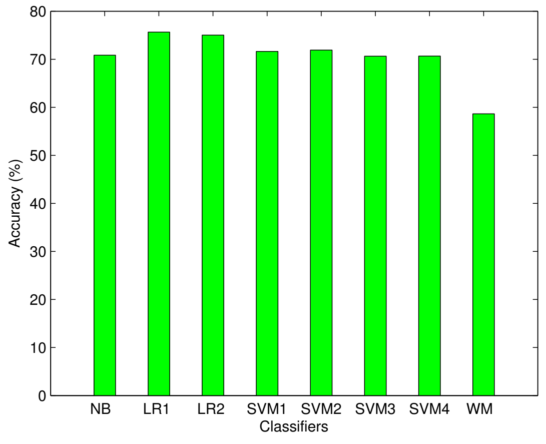
VI. Conclusion

In this paper, we have studied the results of the linear classifiers across different types of data sets. We found that Logistic Regression performs well across all types of data set. An interesting result is that SVMs are better than Logistic Regression for data sets with only numeric attributes. Also, SVMs could perform badly if the C value is not proper. Naive Bayes also performs well across different data sets and though it is not as good as Logistic Regression, its accuracy is comparable to Logistic Regression. Weighted Majority performed poorly across different data sets. We found that the performance of Weighted Majority is highly dependent on the pool of predictors. Also, class imbalance affects the performance of WM if its predictors are affected by it. Unfortunately, the number of data sets we used for ECOC was not very large. In the future, we would like to use more data sets to understand the effectiveness of using ECOC. Also, we would like to do study on data sets which are larger than those that were used here. Finally, we would like to include more linear classifiers for our study.

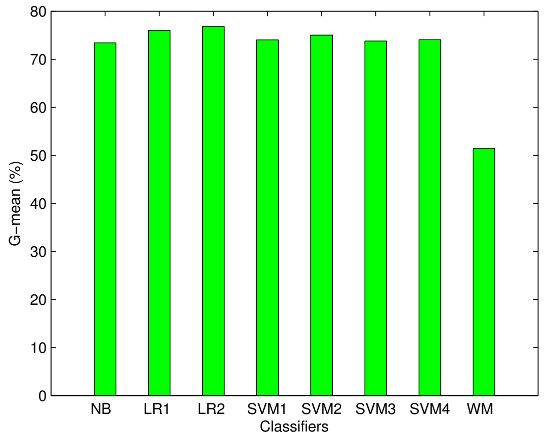
References

[1] M. Galar, A. Fernández, E. B. Tartas, H. B. Sola, and F. Herrera, “An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes,” *Pattern Recognition*, vol. 44, no. 8, pp. 1761–1776, 2011.

[2] D. D. Lewis, “Naive (bayes) at forty: The independence assumption in information retrieval,” in *Machine learning: ECML-98*. Springer, 1998, pp. 4–15.

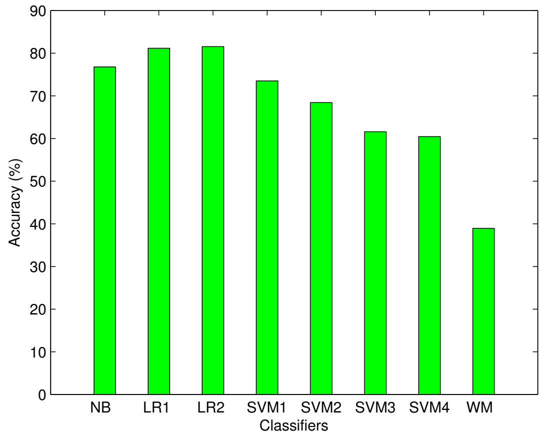


(a) Average Accuracy

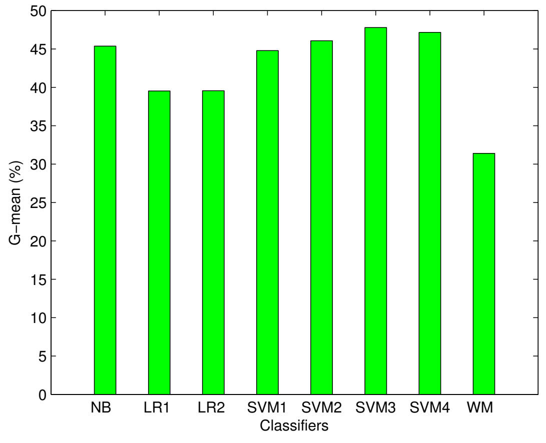


(b) Average G-mean

Figure 5: Results for data sets with both nominal and numerical attributes.



(a) Average Accuracy



(b) Average G-mean

Figure 6: Results for data sets with a large number of instances.

[3] R. Kohavi, "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid." in *KDD*, 1996, pp. 202–207.

[4] I. Rish, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[5] A. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *Advances in neural information processing systems*, vol. 14, p. 841, 2002.

[6] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[7] A. Ben-Hur and J. Weston, "A user's guide to support vector machines," in *Data mining techniques for the life sciences*. Springer, 2010, pp. 223–239.

[8] M. Pal, "Multiclass approaches for support vector machine based land cover classification," *arXiv preprint arXiv:0802.2411*, 2008.

[9] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and computation*, vol. 108, no. 2, pp. 212–261, 1994.

[10] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *arXiv preprint cs/9501101*, 1995.

[11] O. Pujol, P. Radeva, and J. Vitria, "Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 6, pp. 1007–1012, 2006.

[12] J. Friedman, "Another approach to polychotomous classification," Technical report, Stanford University, Department of Statistics, Tech. Rep., 1996.

[13] J. Fürnkranz and E. Hüllermeier, "Pairwise preference learning and ranking," in *ECML*, 2003, pp. 145–156.

[14] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in *NIPS*, 1997.

[15] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification," in *NIPS*, 1999, pp. 547–553.

[16] J. C. Huhn and E. Hüllermeier, "Fr3: A fuzzy rule learner for inducing reliable classifiers," *IEEE T. Fuzzy Systems*, vol. 17, no. 1, pp. 138–149, 2009.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD*

Explorations, vol. 11, no. 1, pp. 10–18, 2009.

[18] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Eleventh Conference on Uncertainty in Artificial Intelligence*. San Mateo: Morgan Kaufmann, 1995, pp. 338–345.

[19] M. Sumner, E. Frank, and M. Hall, "Speeding up logistic model tree induction," in *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*. Springer, 2005, pp. 675–683.

[20] S. le Cessie and J. van Houwelingen, "Ridge estimators in logistic regression," *Applied Statistics*, vol. 41, no. 1, pp. 191–201, 1992.

[21] http://web.engr.oregonstate.edu/ tgd/, last seen: 2014-12-16.

[22] A. Asuncion and D. J. Newman, "Uci machine learning repository," 2007.