IEE 520/BMI 555 Fall 2017 Project

- **NAME: KAKARLA SAI KEERTHY**

ASU ID : 1211263167

Objective:

To use the training data provided and build a classification model with weka that has least balanced error rate and use that model on testing data provided to generate predictions.

Approach:

The given training data file contains 3 classes with type0, type1 and type2 classes having 3694, 1600 and 842 records respectively. Due to the class imbalance problem, we use filters provided under preprocessing tab in weka.

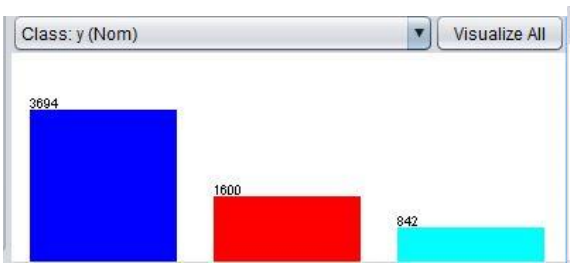
Data preprocessing:

SMOTE is a method which resamples the dataset by applying the Synthetic Minority Oversampling Technique. Go to filter -> filters -> supervised -> instance -> SMOTE.

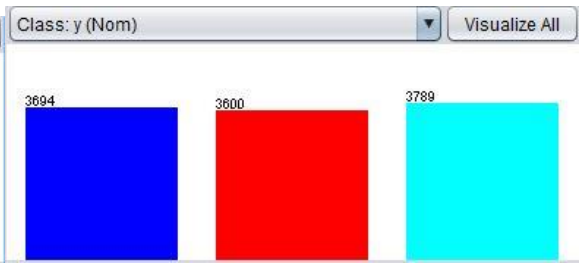
a) **Case1**: Change the classValue to 0 and percentage to 350 under SMOTE to increase the type2 class records from 842 to 3789. Then, change the classValue to 2 and percentage to 125 to change the type1 class instances from 1600 to 3600. Now, the count for each class is close to each other. Then, go to filter -> filters -> unsupervised -> instance -> Randomize to randomize the order of instances in the data file.

b) **Case2**: Change the classValue to 0 and percentage to 300 under SMOTE to increase the type2 class records from 842 to 3368. Then, change the classValue to 2 and percentage to 120 to change the type1 class instances from 1600 to 3519. Then use spreadsubsample filter for undersampling. Now, the count for each class is the same (3368 for each class). Then, go to filter -> filters -> unsupervised -> instance -> Randomize to randomize the order of instances in the data file. In this case, as we can observe from the histogram attached below, we can say that we lose some of the records in type0 class. So, there is a increase in error rate for type 0 classes compared to other two classes.

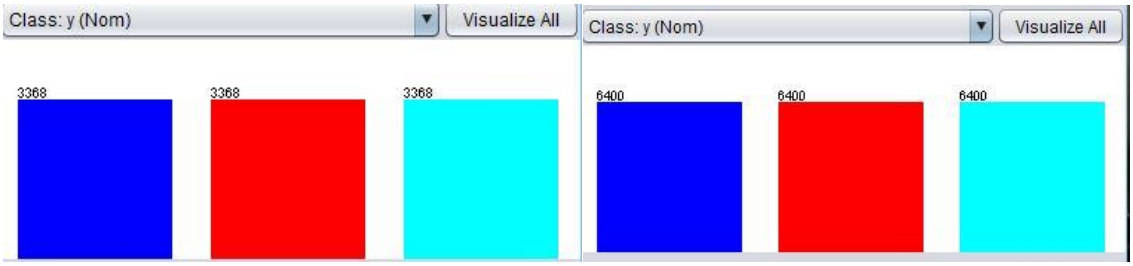
c) **Case3**: Due to the drawback in the above case, in this case we modify the procedure such that SMOTE is applied till the class with majority instances get doubled followed by under sampling using spreadsubsample filter.



a) Histogram of training data provided



b) Histogram for case1 filter



c) Histogram for case2 filter d) Histogram for case3 filter

Filters		Base classifiers	Error% for type0 class	Error% for type1 class	Error% for type2 class	Balanced error rate	Accuracy
SMOTE	with 350% on type2 and 125% on type1	J48	24.6	30.83	18.58	24.67	75.33
		SVM	37.54	23.55	26.76	25.8	74.2
		Random Forest	13.77	26	16.31	18.69	81.31
SMOTE and SpreadSubsample	with 300% on type2 and 120% on type1	J48	26.06	30.58	18.85	25.16	74.84
		SVM	37.67	25.59	30.31	31.19	68.81
		Random Forest	14.69	22.29	14.69	17.22	82.78
SMOTE and SpreadSubsample	Till majority class instances get doubled	***J48***	***21.4***	***17.35***	***8.92***	***15.89***	***84.11***
		SVM	28.56	19.45	17.78	21.93	78.07
		Random Forest	***9.31***	***14.84***	***8.34***	***10.83***	***89.17***

Results of applying filters with j48, svm and random forest classifiers

From the above table, we can see that the balanced error rate is less for 3rd case. So, we select those filter modifications and go to classification methods with different parameter settings.

Note: From observations, bayes classifier, jrip or kNN don't work well for the training data provided, so their results are not tabulated.

Classification Methods and their parameter settings:

Classifiers	Classifier parameters			Error% for type0 class	Error% for type1 class	Error% for type2 class	Balanced error rate	Accuracy
J48		NumofLeaves	SizeofTree	21.48	17.35	8.92	15.91	84.09
	C= 0.25, M = 2	1180	2359					
	C= 0.25, M = 5	631	1261	22.14	18.65	10.12	16.97	83.03
	C= 0.25, M = 10	378	755	21.89	20.15	11.15	17.73	82.27
	c=0.1, M = 2	883	1765	21.15	17.59	9.14	15.96	84.04
	c=0.1, M = 5	530	1059	21.66	18.65	10.40	16.90	83.1
	c=0.1, M = 10	***317***	***633***	***21.33***	***20.39***	***8.07***	***16.59***	***83.41***
	REP-TRUE, c=0.1, M = 10	178	355	22.29	22.34	12.64	19.09	80.91
	REP-TRUE, c=0.1, M = 7	262	523	21.86	22.07	12.34	18.75	81.25
	REP-TRUE, c=0.25, M = 8	244	487	21.97	21.98	12.40	18.78	81.22
Random Forest	Feat= 10, size =100, Iter = 100			10.68	13.57	8.18	10.85	89.15
	Feat = 10, size =75, Iter = 100			10.96	13.70	8.20	10.95	89.05
	Feat = 10, size-75, Iter = 150			10.93	13.75	8.18	10.95	89.05
	Feat = 7, size = 75, It =150			9.89	14.46	6.82	10.39	89.61
	Feat = 30, size = 75, It = 150			13.15	13.23	8.14	11.50	88.5
	Feat = 8, size = 75, Depth = 20			10.53	14.48	8.32	11.01	88.99
	Feat =8, size = 75, Depth = 60			10.48	14.32	8.32	11.04	88.96
	Feat =8, size = 100, depth = 20			***10.35***	***14.32***	***8.37***	***11.01***	***88.99***
	Feat=10, size = 100, depth = 20			10.84	13.87	8.20	10.97	89.03

We know that as Minimum number of objects in j48 classifier increase, the complexity of the tree decreases. As confidence factor decreases, there will be more pruning and size of the tree decrease. So the case with less confidence factor and more minimum number of objects is selected as highlighted in the table above.

For Random Forests, as number of features increase, the classifiers will have more correlation between trees and so it doesn't satisfy the property of independence for it to satisfy binomial probability. If the number of features is less, then the trees built don't give better accurate model. So the value of number of features should be selected carefully. The depth of the trees should be less for a less complex model. So, the case with less depth and a moderate number of features is selected as highlighted in the table above.

Homogeneous Ensemble Methods:

Classifier	Classifier parameters			Error% for type0 class	Error% for type1 class	Error% for type2 class	Balanced error rate	Accuracy
Adaboost with j48		Number of leaves	Size of tree	17.18	14.09	6.95	12.74	87.26
	C= 0.1, M=10	595	1189					
Bagging with j48, It=10	C=0.1,M=10			17.39	16.40	10.01	14.6	85.4
Bagging with j48, It=50	C=0.1,M=10			16.25	15.85	9.76	13.95	86.05
Bagging with RandomForest	Feat =8, size = 100, depth = 20			10.65	15.20	8.67	11.50	88.49
Adaboost with RF	***Feat =8, size = 100, depth = 20***			***10.21***	***15.07***	***8.14***	***11.14***	***88.86***

Note: For Bagging and Adaboost with RF, cross validation folds are reduced to 6, NumOfIterations in RF are 25 and NumOfIterations in Bagging are 5 due to memory limitations of weka (As there are more number of instances in the training data).

For j48 and RF base classifiers with the parameters selected above, bagging and boosting methods are used for improving the accuracy of classification model.

In bagging, as we increase the number of iterations, number of trees built increases and so accuracy increases (For Bagging with J48, NumOfITerations with 10 and CV folds 10, 10*10= 100 trees will be built). But from observations from the above table, Adaboost works better than bagging for this type of training data provided. Also, the advantage of Adaboost is that it builds a low variance, low bias model. So, there will be less chance for overfitting in the model.

Advantage of RF is that each tree is grown fully and averaging is done at the end, so there is less chance for Overfitting. So from the above points and the observations made from the table, we can conclude that Adaboost with RF works better than Adaboost with J48 for the training set provided.

Conclusion:

Adaboost with RF with parameter settings Features=8, Size = 100, Depth = 20 is selected as the final model as it reduces the balanced error rate comparatively well than other models built. We can improve this further with increasing number of Iterations and changing cross validation folds accordingly. It was not performed due to weka memory limitations.