Final Term Project
Fall – 2017

Instructor: Dr. George Runger

Submitted by:
Amrut Deshpande
ASU ID: 1211537636

Objective

To build a classification model for the given data set based on the available classifiers in Weka, to explore the classifier settings and to learn and interpret the classification results.

Brief Data Inference

Weka, a software with the collection of machine learning algorithms for data mining tasks is used for visualizing and interpretation of Data. The dataset contains 41 attributes including one class attribute (y). The training dataset contains 6136 instances. Out of the 41 attributes, one of it is a class attribute which is also a nominal attribute with class labels Class 0, Class 1 and Class 2. Further, during the interpretation class imbalance was observed, where the ratio of majority class to minority class is 4.39. The class 1-3694 instances heavily outnumber instances from the class 0-842 instances whereas Class 2 has 1600 instances.

Pre-processing of Data(Filtering)

Weka provides two types of filters; which are Supervised and Unsupervised. We use Supervised as the unsupervised discretize filter only considers the attribute being discretized. However, the supervised discretize filter will break the attribute into bins that provide the most information about the class. Further, the supervised classifier has five various instance filters like ClassBalancer, Resampling, SMOTE, SpreadSubSample and StratifiedRemoveFolds that can be used to remove the problem of class imbalance.

SMOTE is used for over sampling and SpreadSubSample is used for under sampling. SMOTE was used to up-sample the minority classes to nearing the majority class and SpreadSubsample filter was applied to balance the dataset by making distributionSpread to 1.0. Below, shows the Histogram comparison of the instances imbalance and the instances ratio match obtained by using SMOTE and SpreadSubsample of one of the cases.

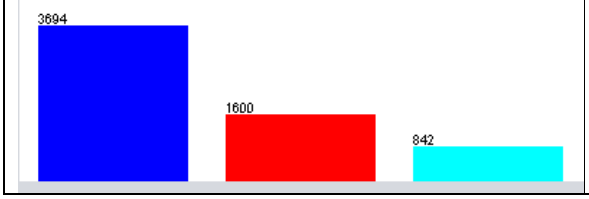
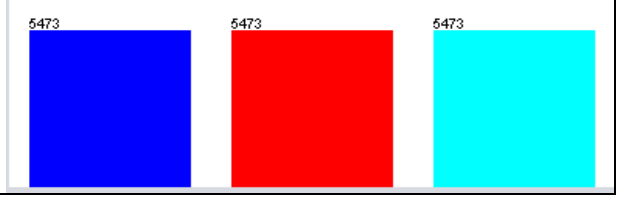
Pre-Process (Class Instances)	Post-Process (Class Instances)
	

Table 1

The three cases of Filtering are explained below:

Case 1: Here SMOTE is used as a filter and further to up sample the minority classes we increase their percentage values. As explained before we obtain class 1 as the majority with 3694 instances and class 0, class 1 have 842 and 1600 instances respectively before SMOTE. We then apply SMOTE twice with percentage parameter increase of 100% to class 0 to up sample it to 3368. Secondly, Class 2 is increased by 125% to obtain 3600 instances.

Filter	Filter Parameter	Classifier	Incorrectly classified	Accuracy	Class 0-Error rate	Class 1-Error rate	Class 2-Error rate	Sum of error rates	Balanced Error rate
SMOTE	Class 0 – 100% *2>Class 2 - 125%	J48	23.35	76.65	0.2467	0.2853	0.1636	0.6958	0.2319
		Bayes Net	29.41	70.59	0	0.5558	0.2604	0.8162	0.2721
		SVM	28.75	71.25	0.3652	0.2222	0.272	0.8594	0.2866
		Random Forest	18.42	81.58	0.1329	0.2517	0.1683	0.5529	0.1843

Table 2

A) **Case 2**: Similar approach of up sampling is used in this case by increasing the percentage of class 0 by 100% thrice followed by class 2 by 15% and finally class 0 by 10%. Here we introduce SpreadSubsample to match all the classes and keep the class instance ratio to 1. We get 3680 instances of each class in this case. But we might miss few true instances of majority class here with the spread subsampling option**.**

Filter	Filter Parameter	Classifier	Incorrectly classified	Accuracy	Class 0- Error rate	Class 1- Error rate	Class 2- Error rate	Sum of error rates	Balanced Error rate
SMOTE	Class 0 – 100% *3>Class 2 - 15%> Class 0 – 10%, SpreadSubsample	J48	22.11	77.89	0.239	0.277	0.147	0.663	0.221
		Bayes Net	29.41	70.94	0	0.5652	0.2383	0.8035	0.2678
		Jrip	23.89	76.11	0.2307	0.2870	0.1992	0.7169	0.2390
		RandomForest	17.69	82.31	0.1329	0.2457	0.1522	0.5308	0.1769

Table 3

B) **Case 3**: This is a crucial case in our pre-processing as this captures all the true instances of majority class. Here we up sample the majority class by 50% to increase its instances to 5541 and further increase class 0 and class 2 by 550% and 250% to increase the instances to 5473 and 5600 respectively. We then use SpreadSubsample to match the class instances ratio to 1 and the final instance count of all three classes will be 5473.

Filter	Filter Parameter	Classifier	Incorrectly classified	Accuracy	Class 0- Error rate	Class 1- Error rate	Class 2- Error rate	Sum of error rates	Balanced Error rate
SMOTE	Class 1 – 50%, Class 0 - 550%>Class 2 - 250%, SpreadSubsample	J48	21.66	78.34	0.2604	0.2414	0.1478	0.6496	0.2165
		Bayes Net	29.76	70.24	0.2722	0.4301	0.1941	0.8964	0.2988
		Jrip	22.37	77.63	0.2333	0.2657	0.1721	0.6711	0.2237
		RandomForest	15.15	84.85	0.1343	0.1911	0.1292	0.4546	0.1515

Table 4

Randomization filter is used after every case so that there is no bias in the interpretation and for the error rate variance to not turn out to be huge. Randomization classifies the data evenly with all the classes placed in equal ration when you randomly pick any subset of data.

Data Classification Methodology

Using the pre-processed data with the filters mentioned above, we go ahead with the classification tools. Currently the data set has uneven class instances for case 1, case 2 and case 3. The motive is to have a model with lower complexity, unbiased class ratio and lower balanced error rate. We also look for models that produce similar number of misclassified instance on each class. This approach makes the predictions unbiased which is appropriate.

While running the dataset on various filter combinations and few base learners, it was found that J48, BayesNet, SVM and RandomForest gave us higher accuracies ranging above 70% which are shown in Table 2,3 and 4. From these tables showing performance of various base learners with the pre-processed data, BayesNet provides us with a biased result with high classification error rate on the majority class compared to other two minor classes. SVM was not a good classifier among all so it wasn't considered for the case 2 and case 3. Instead JRip was tested but the Balanced error rate did not show any kind of improvement.

Thus, we go ahead with two base learners being J48 & RandomForest and the criteria for choosing is Model accuracy, unbiased class error rates and least balanced error rates. The results are tabulated below in Table 4.

Additional tests were done on KNN, NaiveBayes and SVM methods but none were able to give considerable results as all of them had lesser model accuracy and higher balanced error rate. As instructed, only few classifiers tested and tabulated.

Filter	Filter Parameter	Classifier	Incorrectly classified	Accuracy	Class 0-Error rate	Class 1-Error rate	Class 2-Error rate	Sum of error rates	Balanced Error rate
SMOTE	Class 1 – 50%, Class 0 - 550%>Class 2 - 250%, SpreadSubsample	J48	21.66	78.34	0.2604	0.2414	0.1478	0.6496	0.2165
		Bayes Net	29.76	70.24	0.2722	0.4301	0.1941	0.8964	0.2988
		Jrip	22.37	77.63	0.2333	0.2657	0.1721	0.6711	0.2237
		RandomForest	15.15	84.85	0.1343	0.1911	0.1292	0.4546	0.1515

Table 4

Random Forest classifier and J48 both being least error contributors among all classifiers shows that it is best to use Decision trees as our classifier for this data set and to further enhance the model and get a clearer picture of the set accuracy and error rates we dig deep in evaluating the parameter settings of both the classifiers.

Classifier	Classifier Parameter	Number of leaves	Size of the Tree	Accuracy	Class 0-Error rate	Class 1-Error rate	Class 2-Error rate	Sum of error rates
J48	C 0.25, M 2	1141	2281	78.35	0.2604	0.2414	0.1478	0.2165
	C 0.1, M 2	777	1553	79.23	0.2337	0.2430	0.1464	0.2077
	C 0.1, M 3	666	1331	79.18	0.2319	0.2452	0.1476	0.2082
	C 0.1, M 5	462	923	79.09	0.2357	0.2405	0.1511	0.2091
	C 0.1, M 7	368	735	79.34	0.2302	0.2346	0.1549	0.2066
	C 0.1, M 9	305	609	79.11	0.2348	0.2352	0.1568	0.2089
	C 0.1, M 7, REP	224	477	78.34	0.2443	0.2474	0.1582	0.2166
Random Forest	BS=100, NF=0, MD=0			84.85	0.1343	0.1911	0.1292	0.1515
	BS=85, NF=0, MD=0			84.93	0.1321	0.1902	0.1297	0.1507
	BS=70, NF=0, MD=0			84.77	0.1352	0.1904	0.1314	0.1523
	BS=85, NF=6, MD=0			84.93	0.1321	0.1902	0.1297	0.1507
	BS=70, NF=6, MD=0			84.77	0.1352	0.1904	0.1314	0.1523
	BS=85, NF=6, MD=10			82.19	0.2017	0.1886	0.1447	0.1782
	BS=85, NF=6, MD=20			84.62	0.1367	0.1939	0.1308	0.1538
	BS=85, NF=6, MD=30			84.94	0.1323	0.1895	0.1299	0.1506
	BS=85, NF=6, MD=40			84.93	0.1321	0.1902	0.1297	0.1507

Table 5

C- Confidence Factor, M- Minimum number of Objects, REP- Reduced Error Pruning used
BS- Percent Bag Size, NF- Number of Features, MD- Maximum Depth

$$Number\ of\ Features: (\log Classifier/log2) + 1 = (\log 41 / \log 2) + 1 = 6.3576 \sim 6$$

From Table 5, for J48, as we change the ConfidenceFactor 'C' from 0.25 to 0.1, more pruning occurs and reduces the size of the tree without deteriorating the accuracy and the Balanced Error Rate. MinNumObj 'M' was increased by units and observations were made. Complexity of the tree was reduced as we increased the minimum objects. Reduced Error pruning was considered for the

best parameter setting but that did not show any further improvement. The decrease in number of leaves indicate the decrease in the complexity of the tree. Selected parameter settings are highlighted in the table 5 in saffron colour and this decision is made to pick a model with lesser complexity and better-balanced error rate.

For Random forest classifier, The Bag size is reduced to 85 and 70, Number of features was selected based on the formula given in weka=(int(log_2(40) + 1) =6.3576~6(being closest integer). Increasing the depth increases complexity but on the brighter side it reduces the balanced error rate. Thus several iterations were made to identify the change in error rate of the model by increasing the depth. Careful analysis was done to balance both complexity and error rate in the model by picking the highlighted parameters in green of the table 5 as the parameter setting for random forest classifier.

Meta Classifier

The Random Forest model being was selected as Base learner with parameters decided to run AdaBoost and bagging. Meta Classifier like AdaBoost reduces the variance and bias in the model by focusing on previous iterations errors and applying weights on misclassified points. The meta classifiers took large amount of time to get processed in weka, so the CV was reduced to 5 Folds and number of iterations in Random forest was reduced to 50 and meta classifier to 5 iterations.
The results are as shown below in Table 6:

Classifier	Base Classifier parameter	Accuracy	Class O Error rate	Class 1 Error rate	Class 2 Error rate	Balanced Error rate
Adaboost w/ RF	Bag size=85, Max Depth=30, NF=6	84.53	0.1501	0.1893	0.1248	0.1547
Bagging w/RF	Bag size=85, Max Depth=30, NF=6	84.31	0.1403	0.1959	0.1345	0.1569

Table 6

Conclusion and Learning

Adaboost with random forest base classifier is selected as the final classifier model. This model provides better accuracy rate of 84.53 % and a reduced balanced error rate of 0.1547. Selected parameter settings such as bagSizePercent 85, maxDepth 30 and numFeatures 6 controls the complexity of the model without comprising much on accuracy. Although the model without meta classifier slightly has a lesser error rate, still Adaboost is chosen as it overcomes any over fitting in the model. The Selected parameter setting is applied on the test data and the instance result data is submitted as part of the partial completion of project.

References

Introduction to Data Mining, P.-N. Tan, M. Steinbach, V. Kumar, Pearson Addison Wesley, 2006, weka.sourceforge.net, https://www.researchgate.net, https://www.cs.waikato.ac.nz/ml/weka/, https://www.analyticsvidhya.com .