Impact Factor (SJIF): 5.301

International Journal of Advance Research in Engineering, Science & Technology

e-ISSN: 2393-9877, p-ISSN: 2394-2444

Volume 5 , Issue 2, February-2018

Development of Modified Ripper Algorithm to Predict Customer Churn

Abstract: *Technologies such as data warehousing, data mining, and campaign management software have made Customer Relationship Management (CRM) a new area where firms can gain a competitive advantage. Particularly through data mining a process of extracting hidden predictive information from large databases, organisations can identify their valuable customers, predict future behaviors, and enable firms to make proactive, knowledge-driven decisions. Data Mining along with Customer Relationship Management plays a vital role in today's business environment. Customer churn, a process of retaining customer is a major issue. Prevention of customer churn is a major problem because acquiring new customer is more expensive than holding existing customers. In order to prevent churn several data mining techniques have been proposed. One among such method is solving class imbalance which has not received much attention in the context of data mining. This paper describes Customer Relationship Management (CRM), customer churn and class imbalance and proposes a methodology for preventing customer churn through class imbalance.*

Keywords: *Data Mining, Customer Relationship Management, Churn, Class Imbalance.*

1. Introduction

Retail banks often deal with customer churn. Among the several issues addressed by Customer Relationship Management (CRM), identifying the customers who are about to quit the relationship with a company is one of the most important in the financial services industry. When competition becomes tougher, when laws decrease either the barriers to entry or the customer's switching costs, or when a company aims at strengthening its position in a new market, the issue of retaining customers and avoiding customer churn becomes even more crucial. This paper discusses on the design and development of algorithms used in this research and about the banking dataset towards customer churns prediction. The screenshots that have been of the outcome of this research is compared and discussions will be made. The tables are described briefly following with the summary.

2. Case Study - Banking

Acquiring new customers is a more costly process than retaining existing customers. Therefore, the management of relationship with customers plays a vital role in improving the overall profitability of a company. *Churn* is defined as the propensity of a customer to cease doing business with a company in a given time period. This paper emphasize on modeling churn behavior of bank customer. High cost of customer acquisition and customer education requires companies to make large upfront investments on customers. Five main categories can be identified to classify the approaches to the generation of personalized actions. Each category represents a set of homogeneous approaches which can be used to decide what action should be delivered to what customers in a bank. The following approaches clearly illustrate the customer's ideas and their thinking in various aspects. Computational approach includes all those approaches that build a complete model of customers' behaviour, actions, and customers' reactions based on information stored in a data set. These approaches can use both data mining [10] and optimization models [11, 1, 7]. An example of applications in banking and finance is a retail bank which stores the data related to promotion of stocks, and the reactions of customers who might have purchased those stocks or not. The fundamental condition that enables the adoption of these approaches is the completeness of data. A limitation is that only marketing actions already launched before can be considered in such approach. The full coverage of customers may be another problem because some customers' reactions may remain unknown for instance, when a customer does not respond to a survey. Similarity-Based approach is used by Recommender Systems [7] and Web content personalization methods [6]. This kind of approach assumes that actions are related to customer preferences, preferences may be inferred by customer profiles, and that either similar customers behave similarly or similar actions cause similar reactions. A "similarity-based" approach does not require to store as much information as a computational approach. Recording customers' preferences is enough, because it is assumed that the unknown preferences of a customer can be derived by identifying the similarity with other customers. However, the twofold condition of applicability of such approaches is that customers' profiles have to represent preferences, and only actions associated with those preferences can be generated. For instance, a customer who owns multiple credit cards can be classified as a customer who "prefers" using credit cards, whereas the fact that a customer has a mort- gage does not necessarily represent a "preference". A "similarity-based" approach is useful to automate the personalization process. Bottom-Up approach includes the knowledge discovery methods [6, 4] and the use of front office personnel. These approaches consist of two separate steps: 1) pro-filing customers, 2) deciding proper actions, where the first step has to precede the second step (i.e., actions depend on profiles). They cannot be fully made automatic because only the first step is performed by an algorithm. For this reason these approaches are typically not very efficient[5]. The advisor has periodic conversations with a customer, analyzes her needs and proposes tailored financial solutions.Top-Down approache includes the direct marketing approaches [8]. They consist of the same two separate steps typical in bottom-up approaches. However, in this case, the decision of what actions to deliver is made before the definition of customers' profiles and, hence, pro-files depend on actions. For instance, a retail bank managers may

first decide to offer customers a discount on bank transfers, and then select the target customers by building appropriate profiles.

3. Handling Class Imbalance

Six categories of problems that arise when mining imbalanced classes [9].

- Improper evaluation metrics
- Lack of data: absolute rarity
- Relative lack of data
- Data fragmentation
- Inappropriate inductive bias
- Noise

4. Evaluation Metrics

Classification Accuracy is often hard or nearly impossible to construct a perfect classification model that would correctly classify all examples from the test set. Therefore, we have to choose a suboptimal classification model that best suits our needs and works best on our problem domain. In our case, we could use a classifier that makes a binary prediction or a classifier that gives a probabilistic class prediction to which class an example belongs. The first is called binary classifier and the latter is called probabilistic classifier. One can easily turn a probabilistic classifier into a binary one using a certain threshold traditionally so that the Yrate in the test set is equal to the churn rate in the original training set. Binary Classifiers always label one class as a positive (in our case a churner) and the other one as a negative class (a non churner). The test set consists of P positive and N negative examples. A classifier assigns a class to each of them, but some of the assignments are wrong. To assess the classification results we count the number of true positive (TP), true negative (TN), false positive (FP) (actually negative, but classified as positive) and false negative (FN) (actually positive, but classified as negative) examples. It holds

$$TP+FN=P$$
and
$$TN+FP=N$$

The classifier assigned TP + FP examples to the positive class and TN + FN examples to the negative class. Let us define a few well known and widely used measures:

$$specificity = \frac{TN}{N} \Rightarrow 1 - \frac{FP}{N} = FPrate$$

$$sensitivity = \frac{TN}{P} = FPrate = recall$$

$$Yrate = \frac{TP + FP}{P + N}$$

$$precision = \frac{TP}{TP + FP}$$

$$Accuracy = \frac{TP + TN}{P + N}$$

$$\Rightarrow misclassifcation\ error\ (MER) = 1 - accuracy$$

Precision, recall and accuracy (or MER) are often used to measure the classification quality of binary classifiers. The FPrate measures the fraction of non churners that are misclassified as churners. The TPrate or recall measures the fraction of churners correctly classified. Precision measures that fraction of examples classified as churner that are truly churner. Area under ROC curve is often used as a measure of quality of a probabilistic classifier. It is close to the perception of classification quality that most people have. AUC is computed with the following formula:

$$AUC = \int_0^1 \frac{TP}{P} d\frac{FP}{N} = \frac{1}{P.N} \int_0^N TP dFP$$

For each negative example count the number of positive examples with a higher assigned score than the negative example, sum it up and divide everything with P * N. This is exactly the same procedure as used to compute the probability that a random positive example has a higher assigned score than random negative example.

$$AUC = P(Score_{Random\ churner} > Score_{Random\ non\ churner})$$

In many data mining tasks, including churn prediction, it is the rare cases that are of primary interest. Metrics that do not take this into account generally do not perform well in these situations. One solution is to use cost-sensitive learning methods [3, 10]. Table 1 shows the performance of various existing algorithms such as decision trees, gradient boosting and weighted random forest on random over sampling method. From the results of performance metric area under curve it is clear that

decision tree performs better than the other algorithms such as decision tree and weighted random forest algorithms. It is clear that in almost all iterations from 10 to 100, the performance AUC is better in decision tree. This table also depicts the error performance of various existing algorithms such as decision trees, gradient boosting and weighted random forest on random over sampling method. From the results of performance metric error it is clear that gradient boosting reduces the error rate than the other algorithms such as decision tree and weighted random forest. Also, it is clear the performance error rate is reduced in gradient boosting method. The table shows the performance accuracy of the existing algorithms. From all the iterations it can be understood that the gradient boosting algorithm performs better than that of other algorithms. Graphical representation of the comparison is given in Fig. 1. Table 2 shows the performance of various existing algorithms such as decision trees, gradient boosting and weighted random forest on random over sampling method. From the results of performance metric area under curve it is clear that weighted random forest performs better than the other algorithms such as decision tree and gradient boosting. It is clear that in almost all iterations from 10 to 100, the performance AUC is better in weighted random forest. This table also depicts the error performance of various existing algorithms such as decision trees, gradient boosting and weighted random forest on random over sampling method. From the results of performance metric error it is clear that gradient boosting reduces the error rate than the other algorithms such as decision tree and weighted random forest. Also, it is clear that in almost all iterations from 10 to 100, the performance error rate is reduced in gradient boosting method. The table shows the performance accuracy of the existing algorithms. From all the iterations it can be understood that the gradient boosting algorithm performs better than that of other algorithms. Graphical representation of the comparison is given in Fig. 1. Table 3 shows the performance of various existing algorithms such as decision trees, gradient boosting and weighted random forest on random over sampling method. From the results of performance metric area under curve it is clear that weighted random forest performs better than the other algorithms. It is clear that in almost all iterations from 10 to 100, the performance AUC is better in gradient boosting method. This table also depicts the error performance of various existing algorithms such as decision trees, gradient boosting and weighted random forest on random over sampling method. From the results of performance metric error it is clear that gradient boosting reduces the error rate than the other algorithms such as decision tree and weighted random forest. Also, it is clear that in almost all iterations from 10 to 100, the performance error rate is reduced in gradient boosting method. The table shows the performance accuracy of the existing algorithms. From all the iterations it can be understood that the gradient boosting algorithm performs better than that of other algorithms. Graphical representation of the comparison is given in Fig. 1. Table 4 shows the performance of various proposed algorithms such as genetic algorithm, ripper algorithm and k-nearest neighbor algorithm on random over sampling method. From the results of performance metric area under curve it is clear that genetic algorithm gives better performance than the ripper algorithm and k-nearest neighbor algorithm. It is clear that in almost all iterations from 10 to 100, the performance AUC is better in both genetic algorithm and k-nearest neighbor algorithm. This table also depicts the error performance of various proposed algorithms such as genetic algorithm, ripper algorithm and k-nearest neighbor algorithm. From the results of performance metric error it is clear that genetic algorithm and ripper algorithm reduces the error rate than the other algorithm namely k-nearest neighbor. Also, it is clear that in almost all iterations from 10 to 100, the performance error rate is reduced in genetic algorithm and ripper algorithm. The table shows the performance accuracy of the proposed algorithms. From all the iterations it can be understood that the genetic algorithm and ripper algorithm performs better than that of other algorithm. Graphical representation of the comparison is given in Fig. 2. Table 5 shows the performance of various proposed algorithms such as genetic algorithm, ripper algorithm and k-nearest neighbor algorithm on random under sampling method. From the results of performance metric area under curve it is clear that genetic algorithm is giving better performance than the k-nearest neighbor algorithm and ripper algorithm. It is clear that in almost all iterations from 10 to 100, the performance AUC is better in genetic algorithm. This table also depicts the error performance of various proposed algorithms such as genetic algorithm, ripper algorithm and k-nearest neighbor algorithm. From the results of performance metric error it is clear that ripper algorithm reduces the error rate than the other algorithms namely k-nearest neighbor and genetic. Also, it is clear that in almost all iterations from 10 to 100, the performance error rate is reduced in ripper algorithm. The table shows the performance accuracy of the proposed algorithms. From all the iterations it can be understood that the ripper algorithm performs better than that of other two algorithms. Graphical representation of the comparison is given in Fig 2. Table 6 shows the performance of various proposed algorithms such as genetic algorithm, ripper algorithm and k-nearest neighbor algorithm on advanced random under sampling method. From the results of performance metric area under curve it is clear that genetic algorithm is giving better performance than the k-nearest neighbor algorithm and ripper algorithm. It is clear that in almost all iterations from 10 to 100, the performance AUC is better in genetic. This table also depicts the error performance of various proposed algorithms such as genetic algorithm, ripper algorithm and k-nearest neighbor algorithm.

Table 1. Random Over Sampling for DT, GB and WRF algorithms.

Algorithm \ Metrics		10	20	30	40	50	60	70	80	90	100
DT	AUC	0.0080	0.0363	0.0960	0.1473	0.2355	0.3040	0.4281	0.5633	0.7182	0.9086
	ERROR	0.2500	0.2632	0.3484	0.3625	0.3643	0.3800	0.4000	0.4080	0.4286	0.4308
	ACCU	56.9231	57.1429	59.2000	60.0000	62.0000	63.5714	63.7500	65.1613	73.6842	75.0000
GB	AUC	0.0114	0.0416	0.0929	0.1480	0.2365	0.3116	0.4298	0.5690	0.7171	0.9082
	ERR	0.1600	0.1760	0.1274	0.1353	0.1229	0.1399	0.1740	0.1587	0.1435	0.1324
	ACC	83.1579	84.5455	84.6154	85.6000	85.7143	85.7143	6.2500	87.0968	88.0000	90.0000
WRF	AUC	0.0142	0.0550	0.0878	0.6313	0.2960	0.4221	0.4645	0.4635	0.8300	0.8703
	ERR	0.2000	0.2960	0.3000	0.3077	0.3429	0.3800	0.4065	0.4250	0.4842	0.50710
	ACC	49.2857	51.5789	57.5000	59.3548	62.0000	65.7143	69.2308	70.0000	70.4000	80.0000

Table 2. Random Under Sampling for DT, GB and WRF algorithms.

Algorithm \ Metrics		**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
DT	AUC	0.0242	0.0722	0.1417	0.2558	0.3834	0.5654	0.7798	1.0018	1.2582	1.05654
	ERROR	0.1600	0.2000	0.2162	0.2207	0.2222	0.2381	0.2424	0.2488	0.2769	0.2824
	ACC	71.7647	72.3077	75.1220	75.7576	76.1905	77.7778	77.9310	78.3784	80.0000	84.0000
GB	AUC	0.0234	0.0750	0.1479	0.2572	0.3822	0.5711	0.7809	1.0072	1.2600	1.5633
	ERR	0.0800	0.1333	0.1366	0.1379	0.1394	0.1405	0.1440	0.1529	0.1538	0.1619
	ACC	72.8295	84.7354	84.5965	85.5300	85.3299	86.8606	86.7869	86.3285	86.3427	90.0000
WRF	AUC	0.3112	0.0760	0.0652	0.3121	0.7600	0.3320	0.5461	0.9124	0.8100	0.5210
	ERR	0.5600	0.5714	0.5727	0.6000	0.6357	0.6800	0.6947	0.7077	0.7625	0.8000
	ACC	20.0000	23.7500	29.2308	30.5263	32.0000	36.4286	40.0000	42.7273	42.8571	44.0000

Table 3. Advanced Random Under Sampling for DT, GB and WRF algorithms.

Algorithm \ Metrics		**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
DT	AUC	0.0251	0.0676	0.1494	0.2517	0.3819	0.5683	0.7740	0.9999	1.2617	1.5632
	ERROR	0.1200	0.1440	0.1448	0.1512	0.1556	0.1568	0.1576	0.1692	0.1765	0.1810
	ACCU	81.9048	82.3529	83.0769	84.2424	84.3243	84.4444	84.8780	85.5172	85.6000	88.0000
GB	AUC	0.0247	0.0726	0.1476	0.2579	0.3831	0.5684	0.7797	1.0059	1.2591	1.5637
	ERR	0.0800	0.1333	0.1366	0.1379	0.1394	0.1405	0.1440	0.1529	0.1538	0.1619
	ACC	83.8095	84.6154	84.7059	85.6000	85.9459	86.0606	86.2069	86.3415	86.6667	92.0000
WRF	AUC	0.3212	0.0542	0.6768	0.1863	0.4380	0.1311	0.5405	0.3225	0.8430	0.9430
	ERR	0.1000	0.1200	0.1290	0.1429	0.1429	0.1440	0.1500	0.1538	0.1545	0.1684
	ACC	83.1579	84.5455	84.6154	85.0000	85.6000	85.7143	85.7143	87.0968	88.0000	90.0000

Table 4. Random Over Sampling for GA, RA and k-NN algorithms

Algorithm \ Metrics		**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
GA	AUC	0.7331	0.4315	0.3666	0.3594	0.2273	0.0295	0.1352	0.1785	0.5627	0.8204
	ERROR	0.1000	0.1440	0.1545	0.1548	0.1571	0.1579	0.1714	0.1750	0.2000	0.6308
	ACCU	36.9231	80.0000	82.5000	82.8571	84.2105	84.2857	84.5161	84.5455	85.6000	90.0000
RA	AUC	0.7632	0.05476	0.7500	0.3752	0.2863	0.2300	0.5295	0.4215	0.6000	0.8980
	ERR	0.1000	0.1440	0.1545	0.1548	0.1571	0.1579	0.1714	0.1875	0.2000	0.2000
	ACC	80.0000	80.0000	81.2500	82.8571	84.2105	84.2857	84.5161	84.5455	85.6000	90.0000
KNN	AUC	0.3222	0.0520	0.0670	0.4252	0.6613	0.7600	0.3425	0.7625	0.6432	0.7481
	ERR	0.1579	0.1600	0.1625	0.1677	0.1692	0.1786	0.1818	0.2000	0.2500	0.2857
	ACC	71.4286	75.0000	80.0000	81.8182	82.1429	83.0769	83.2258	83.7500	84.0000	84.2105

Table 5. Random Under Sampling for GA, RA and k-NN algorithms

Algorithm \ Metrics		10	20	30	40	50	60	70	80	90	100
GA	AUC	0.0068	0.0418	0.0744	0.1304	0.2079	0.3139	0.4248	0.5648	0.6962	0.8559
	ERROR	0.1440	0.1545	0.1548	0.1571	0.1579	0.1714	0.1875	0.2000	0.2000	0.7500
	ACCU	25.0000	80.0000	80.0010	81.2500	82.6571	84.2105	84.4357	84.5114	84.5321	85.6043
RA	AUC	0.0329	0.4171	0.0870	0.4241	0.3212	0.3862	0.8436	0.4203	0.1417	0.8743
	ERR	0.1000	0.1450	0.2345	0.1431	0.1471	0.2119	0.2014	0.2275	0.2130	0.2980
	ACC	81.0400	80.4400	81.2650	82.8565	84.2435	83.5557	84.4161	84.5675	85.6110	90.0110
KNN	AUC	0.2431	0.0450	0.9700	0.3252	0.3313	0.3540	0.4376	0.5025	0.6652	0.8544
	ERR	0.3219	0.1455	0.1067	0.1520	0.1438	0.1321	0.1870	0.2011	0.2430	0.2865
	ACC	71.4286	75.0000	80.0000	80.0000	81.2903	81.8182	83.0769	83.2000	83.7500	84.2105

Table 6. Advanced Random Under Sampling for GA, RA and k-NN algorithms

Algorithm \ Metrics		10	20	30	40	50	60	70	80	90	100
GA	AUC	0.0098	0.0353	0.0773	0.1287	0.2020	0.3166	0.4290	0.5724	0.6958	0.8516
	ERROR	0.1636	0.1750	0.1800	0.2000	0.2000	0.2065	0.2160	0.2786	0.3000	0.3429
	ACCU	65.7143	70.0000	72.1429	78.4000	79.3548	80.1000	80.0000	82.2100	82.5000	83.6364
RA	AUC	0.0114	0.0488	0.1061	0.1852	0.2690	0.3911	0.5402	0.6715	0.8408	1.0497
	ERR	0.0523	0.0532	0.0569	0.0600	0.0633	0.0642	0.0857	0.0870	0.1000	.2308
	ACC	76.9231	90.0000	91.3043	91.4286	93.5780	93.6709	94.0000	94.3089	94.6809	94.7712
KNN	AUC	0.0172	0.3450	0.0610	0.1242	0.2313	0.2200	0.3321	0.5224	0.6352	0.8861
	ERR	0.1560	0.1240	0.1563	0.1468	0.1779	0.1414	0.1744	0.2110	0.2300	0.2515
	ACC	73.7500	80.0000	80.0000	82.8571	82.8571	84.2105	84.5161	84.5455	85.6000	90.0000

Table 7. Advanced Random Under Sampling for Ripper Algorithm and Modified Ripper Algorithm

Algorithm \ Metrics		10	20	30	40	50	60	70	80	90	100
RA	AUC	0.0114	0.0488	0.1061	0.1852	0.2690	0.3911	0.5402	0.6715	0.8408	1.0497
	ERR	0.0523	0.0532	0.0569	0.0600	0.0633	0.0642	0.0857	0.0870	0.1000	.2308
	ACC	76.9231	90.0000	91.3043	91.4286	93.5780	93.6709	94.0000	94.3089	94.6809	94.7712
MRA	AUC	0.0151	0.0465	0.1070	0.1812	0.2716	0.3913	0.5408	0.6697	0.8410	1.0501
	ERR	0 .0459	0.0523	0.0532	0.0569	0.0580	0.0600	0.0615	0.0633	0.0857	0.1000
	ACC	90.0000	91.4286	93.6709	93.8462	94.0000	94.2029	94.3089	94.6809	94.7712	95.4128

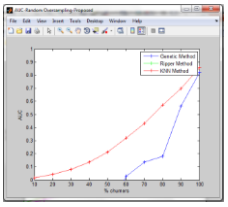
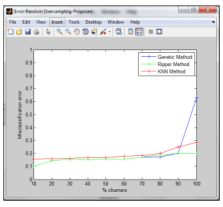
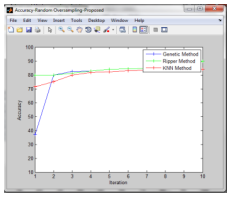
	AUC	ERROR	ACCURACY
ARUS			

Figure 3. Comparison of ripper and modified ripper algorithm with advanced random under sampling

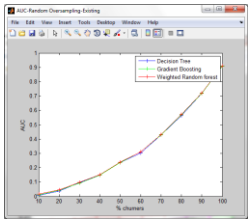
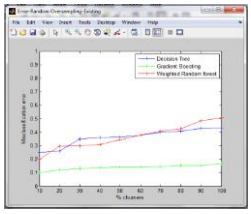
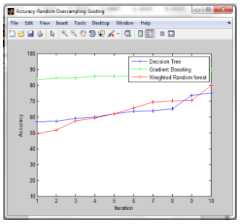
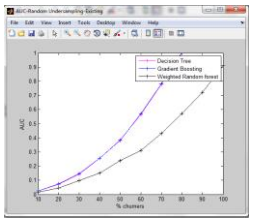
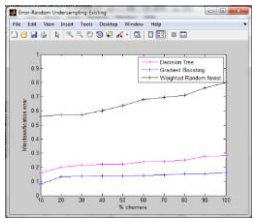
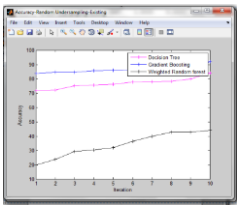
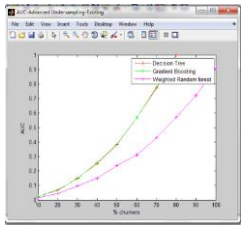
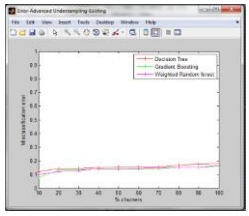
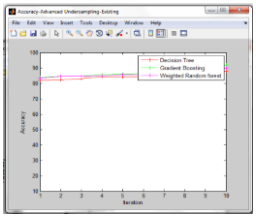
	AUC	ERROR	ACCURACY
ROS			
RUS			
ARUS			

Figure 1. Comparison of DT, GB and WRF algorithms in different sampling method using the AUC, Error and Accuracy

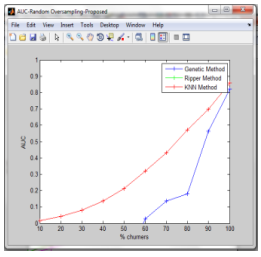
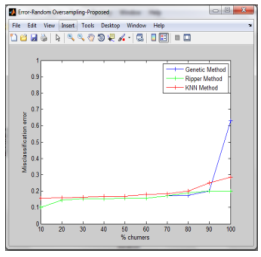
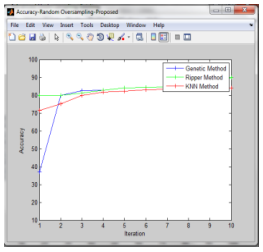
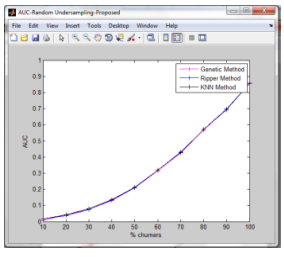
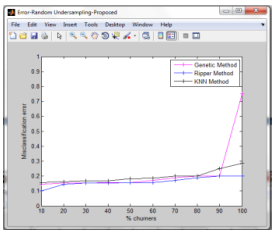
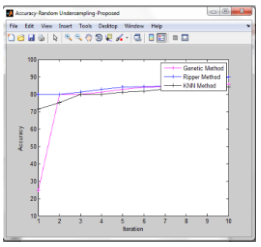
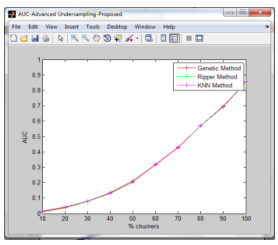
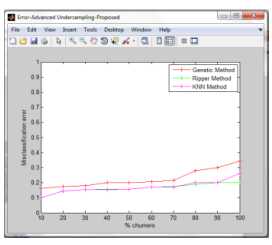
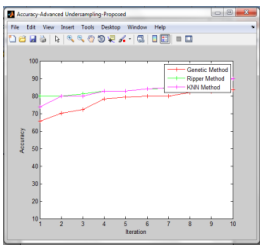
	AUC	ERROR	ACCURACY
ROS			
RUS			
ARUS			

Figure 2. Comparison of GA, RA, k-NN algorithms in different sampling method using the AUC, Error and Accuracy

From the results of performance metric error it is clear that ripper algorithm reduces the error rate than the other algorithms namely k-nearest neighbor and genetic. Also, it is clear that in almost all iterations from 10 to 100, the performance error rate is reduced in ripper algorithm. The table shows the performance accuracy of the proposed algorithms. From all the iterations it can be understood that the k-nearest neighbor algorithm performs better than that of other two algorithms. Graphical representation of the comparison is given in Fig. 2. Table 7 give one step more information about the comparison of ripper algorithm and modified ripper algorithm in terms of area under curve, error and accuracy during the different iterations in sampling methods which include advanced random undersampling method CUBE is used. Based on CUBE it is obtained that the inclusion probabilities from modified ripper algorithm iterations are accurately satisfied. Graphical representation of the comparison between ripper algorithm and modified ripper algorithm is given in Fig. 3.

5. Conclusion

The study thus predicts the churn of customers in banking sector and can then be extended, thereby helping formulate intervention strategies based on churn prediction to reduce the lost revenue by increasing customer retention. It is expected that, with a better understanding of these characteristics, bank managers can develop a customized approach to customer retention activities within the context of their Customer Relationship Management efforts. This paper discussed the development of the Modified Ripper Algorithm and also introduced about the case study on banking sector and banking dataset towards customer churn prediction. The screenshots and results obtained out of the software prototype are tabulated and compared.

References

[1] Ansari and Mela, "E-Customization", Journal of marketing research, Vol. 40, No. 2, 2003, pp. 131-145.
[2] Chen, Liaw, Breiman, "Using random forests to learn imbalanced data". Technical Report 666, Statistics Department, University of California at Berkeley. 2004.
[3] Freund and Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting". Journal of Computer and System Sciences, 55(1), 119–139. 1997.
[4] Kumar, "From Mass Customization to Mass Personal-ization: A Strategic Transformation," International Jour-nal of Flexible Manufacturing Systems, Vol. 19, No. 4, 2007, pp. 533-547.
[5] Linden, Smith and York, "Amazon.com Recom-mendations: Item-to-Item Collaborative Filtering," IEEE Internet Computing, Vol. 7, No. 1, 2003, pp. 76-80.
[6] Mobasher, Berendt and Spiliopoulou, "Knowl-edge Discovery and Data for Personalization," Tutorial at the 12th European Conference on Machine Learning, Freiburg, 5-7 September 2001.
[7] Pancras and Sudhir, "Optimal Marketing Strategies for a Customer Data Intermediary," Journal of Marketing Research, Vol. 44, No, 4, 2007, pp. 560-578.
[8] Reinartz and Venkatesan, "Decision Models for Customer Relationship Management," In: B. Wierenga, Ed., Handbook of Marketing Decision Models, Springer Netherlands, Dordrecht, 2009.
[9] Weiss.G.M, "Mining with rarity: A unifying framework", ACM SIGKDD Explor. Newslett., vol. 6, no. 1, pp. 7–19, 2004.
[10] Wang and Jiang, "Mining Actionable Patterns by Role Models," IEEE International Conference on Data Engineering, Atlanta, 3-7 April 2006, pp. 16-25.
[11] Zhang and Wedel, "The Effectiveness of Custom-ized Promotions in Online and Offline Stores," Journal of Marketing Research, Vol. 46, No. 2, 2009, pp. 190-206.

Author Biography

Mrs M Rajeswari her post graduate degree in Computer Science from Pioneer College of Arts and Science in Coimbatore, India in 2008 and currently pursuing Ph.D. in Computer Science at Bharathiar University, India. She has research interests include Data mining and Customer Relationship Management.