A simulation study on the properties of KCV-SMOTE and KFS with application to credit card fraud prediction

Pao Zhu Vivian Hsu

Student number: 400547994

April 23, 2024

1 Introduction

Credit cards are an electronic form of payment that are popular for the convenience and protection they offer on everyday purchases. Due to its popularity, detecting credit card fraud is crucial for financial companies to maintain customer satisfaction and minimize losses. Machine learning techniques have become the predominant method to predict credit card fraud over the years and companies are constantly looking for ways to improve their predictions.

To this date, a variety of machine learning techniques are being used to predict credit card fraud. Some techniques mentioned in recent literature include random forest, neural networks, SVM, k-nearest neighbours, logistic regression, and Naïve Bayes [4, 1].

As described by Sulaiman et al. [4], researchers often have challenges with the imbalance of fraudulent and genuine transactions when it comes to studying credit card fraud. Kang and Zhang [2] introduced the K-fold cross-validation and synthetic minority oversampling technique (KCV-SMOTE) and key feature scanning (KFS) method to address the challenges posed by unbalanced datasets, specifically within the context of credit card fraud detection. The method was applied to a logistic regression approach and the final model had an AUC of 98.62% [2].

While this adaptation presents a promising approach to improving model performance for imbalanced data, its efficacy across various signal-to-noise ratio (SNR) conditions remains insufficiently explored. Further investigation into how this method performs under varying ratios of SNR predictors would provide valuable insights into its robustness and applicability to a broader range of datasets. Therefore, in this paper, we will investigate the KCV-SMOTE and KFS method's predictive performance under different proportions of unbalanced data and ratios of signal-to-noise.

2 Methods

To simulate the data in this study, we utilized the transaction data simulator developed by the Machine Learning Group of ULB, the Université Libre de Bruxelles [3]. This simulator is based off a public data set on credit card fraud collected by the ULB and Worldline [5], which is frequently used in credit card fraud research including the work done by Kang and Zhang [2, 1].

	CUSTOMER_ID	x_customer_id	y_customer_id	mean_amount	std_amount	mean_nb_tx_per_day	available_terminals	nb_terminals
0	0	54.881350	71.518937	62.262521	31.131260	2.179533	[29, 87, 144, 241, 330, 858, 996, 1028, 1067, ...	78
1	1	42.365480	64.589411	46.570785	23.285393	3.567092	[5, 160, 242, 378, 431, 475, 571, 762, 876, 93...	85
2	2	96.366276	38.344152	80.213879	40.106939	2.115580	[316, 406, 447, 523, 968, 1200, 1318, 1365, 16...	70
3	3	56.804456	92.559664	11.748426	5.874213	0.348517	[65, 94, 113, 364, 401, 433, 485, 651, 672, 77...	70
4	4	2.021840	83.261985	78.924891	39.462446	3.480049	[372, 614, 774, 1362, 1446, 1564, 1637, 1939, ...	65

(a) Customer profile table

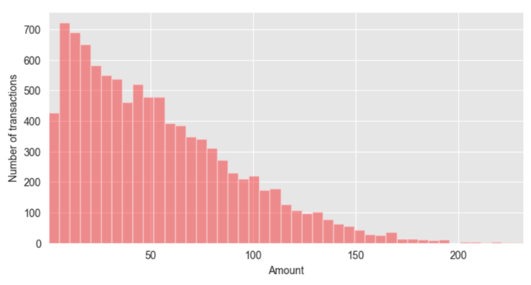
	TERMINAL_ID	x_terminal_id	y_terminal_id
0	0	41.702200	72.032449
1	1	0.011437	30.233257
2	2	14.675589	9.233859
3	3	18.626021	34.556073
4	4	39.676747	53.881673

(b) Terminal profile table

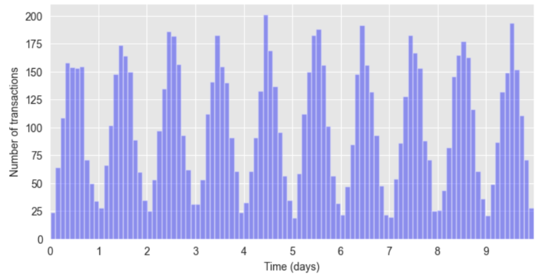
Figure 1: First 5 rows of profile tables

We first started by simulating a large dataset of credit card transactions as the baseline for each of our study cases. To do so, we created 5000 customer profiles and 10,000 terminal profiles. The customer profiles describe a customer's spending habits, where they are located, and what terminals are near them while the terminal profiles describe where the terminal machines are located. Figure 1 provides a glimpse of these profile tables.

From these profiles, we generated about 3.5 million rows of transactions and 7 variables including transaction ID, customer ID, terminal ID, transaction amount, and a few measures on transaction time. These variables were simulated using common probability distributions as illustrated in Figure 2. For example, transaction amounts are primarily simulated using a Normal distribution with mean and standard deviation derived from the spending habits from the customer profiles.



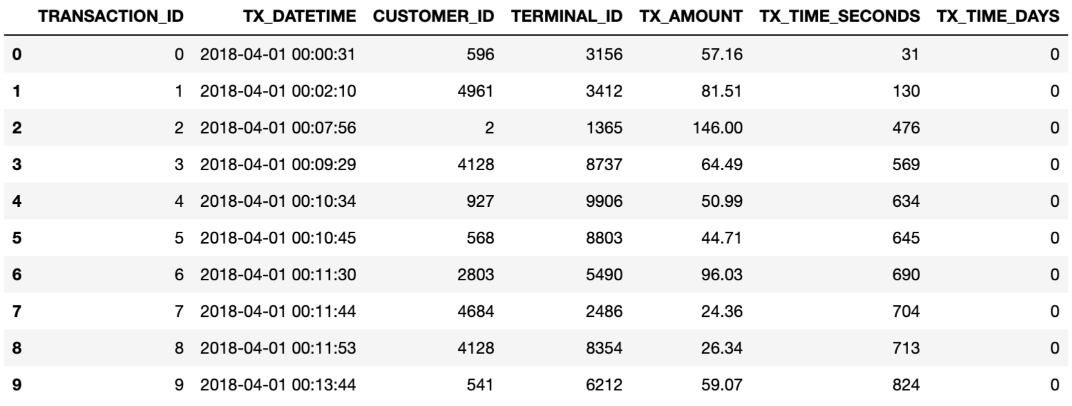
(a) Distribution of amounts. Amounts are simulated using $N(\mu, \sigma)$, but uses $Uniform(0, 2\mu)$ if the amount < 0.



(b) Distribution of days. Time in seconds are simulated $N(86400/2, 20000)$ and converted to days through division.

Figure 2: Distribution of transactions. Let μ and σ be the mean and standard deviation of the spending amount obtained from customer profile respectively.

Transactions start in April 2018 and cover the span of a year. All transactions are considered genuine. Figure 3 shows the first 10 rows of this dataset.



	TRANSACTION_ID	TX_DATETIME	CUSTOMER_ID	TERMINAL_ID	TX_AMOUNT	TX_TIME_SECONDS	TX_TIME_DAYS
0	0	2018-04-01 00:00:31	596	3156	57.16	31	0
1	1	2018-04-01 00:02:10	4961	3412	81.51	130	0
2	2	2018-04-01 00:07:56	2	1365	146.00	476	0
3	3	2018-04-01 00:09:29	4128	8737	64.49	569	0
4	4	2018-04-01 00:10:34	927	9906	50.99	634	0
5	5	2018-04-01 00:10:45	568	8803	44.71	645	0
6	6	2018-04-01 00:11:30	2803	5490	96.03	690	0
7	7	2018-04-01 00:11:44	4684	2486	24.36	704	0
8	8	2018-04-01 00:11:53	4128	8354	26.34	713	0
9	9	2018-04-01 00:13:44	541	6212	59.07	824	0

Figure 3: First 10 rows of the baseline transactions dataset

With the baseline dataset, we then added fraud transactions to the dataset where a fraudulent transaction is indicated by a value of 1 and a genuine transaction is indicated by 0. To study how unbalanced classes impact prediction performance, we created five sets of data where the proportions of fraud transactions are approximately 0.35%, 0.5%, 0.8%, 1%, and 1.5% of the full dataset. These proportions were selected to mimic the proportion of fraud in real data, which tends to be less than 1% [3].

The transaction data simulator creates data simulations based off three scenarios, two of which are based on real credit risk scenarios and one of which is arbitrary and used for testing purposes [3]. In attempt to simulate data most similar to real data, we have modified the simulator so that it can be based on only the real credit risk scenarios. Trial and error was used to tune the parameter inputs so that the proportion of fraud transactions best match those specified earlier. We summarize the parameter tuning in Table 1 below.

Table 1: Parameter tuning for each of the class imbalance datasets

Dataset	Fraud proportion	Number of compromised terminals per day	Number of compromised customers per day
1	0.35%	1	1
2	0.50%	1	3
3	0.80%	2	3
4	1.00%	3	3
5	1.00%	5	3

Once fraudulent transactions were added, we then applied data transformation to add more features to the data such as an indicator on whether the transaction was made during a weekday and an indicator on whether the transaction was made during the night.

To study how noisy data impacts prediction performance, we simulated nineteen more sets of data. Table 2 provides a summary on how we have configured each of these datasets. We created four groups of datasets with fixed fraud proportions to ease the comparison between differing SNRs.

Table 2: Simulated datasets for SNR analysis

Group	Fraud proportion	SNRs
1	1.00%	1:9, 3:7, 5:5, 7:3, 9:1
2	0.80%	1:9, 3:7, 5:5, 7:3, 9:1
3	0.35%	1:9, 2:8, 3:7, 4:6, 5:5, 6:4
4	1.50%	1:9, 5:5, 9:1

Defining signal versus noise predictors required a few assumptions. Firstly, all variables from the baseline dataset were assumed to be signal predictors of fraud. As such, signal predictors were chosen by randomly selecting variables from the baseline dataset. Secondly, all noise predictors were to be simulated in a way that would not relate to existing variables. This was done by assigning a random integer between 1 and 100 inclusive for each of the noise predictors, where there is a 1% probability of assigning each integer.

Once the datasets were established, we then applied Kang and Zhang's [2] KCV-SMOTE and KFS method as outlined in the steps below. An illustration of this process is available in Figure 4 as provided by Kang and Zhang [2].

1. Split the data into a training set and a testing set.

2. Use KCV-SMOTE on the training set to obtain a synthetic training set.

3. Perform KFS on the synthetic training set to obtain a sub-training set.

4. Train the logistic regression classifier for each sub-training set separately and get a set of area under the precision-recall curve (AUROC) scores. Sort the AUROC values and select the sub-training sets with the highest scores.

5. Train the intersection of the sub-training sets from the previous step to obtain the best model, and input the test set into the best model to obtain the final prediction result.

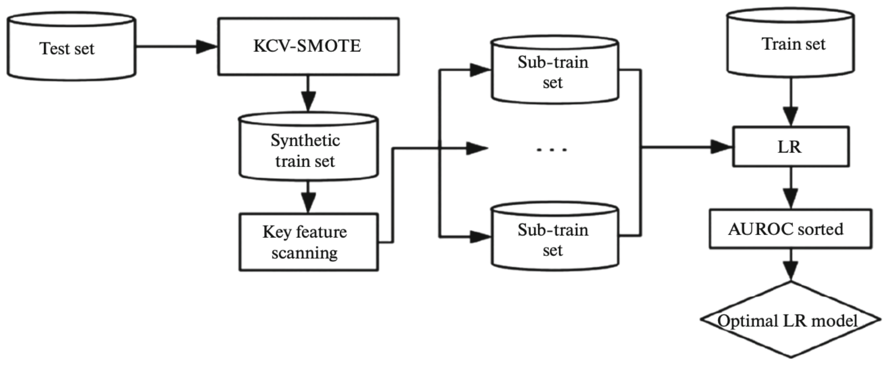


Figure 4: KCV-SMOTE and KFS process proposed by Kang and Zhang [2]

When splitting the data into training and testing sets, we allocated 80% of the data for training and 20% for testing. We then applied KCV-SMOTE to the training set. We divided the dataset into $k = 5$ parts, selected a random part of the dataset to serve as the sub-test set, and created synthetic datasets from each of the remaining parts through oversampling. Logistic regression models were built for each of the parts and the synthetic dataset with the highest resulting AUROC value was selected as the sub-training set.

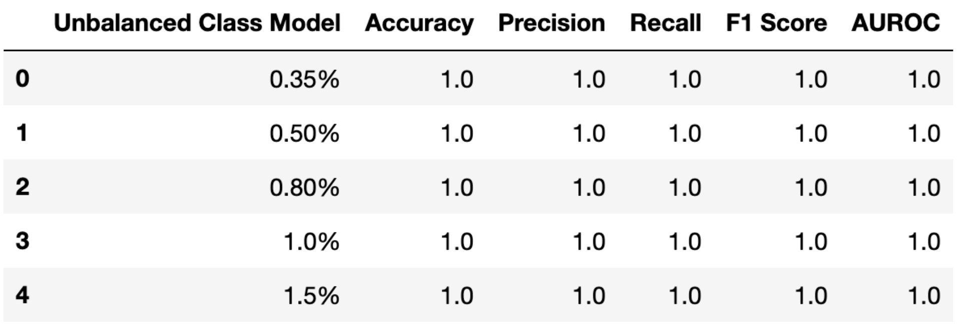
Next, we applied KFS on the sub-training set. Apart from the response variable, we used $k = 0$ key features. We chose to include no key features we have no prior knowledge of what variables are considered important. This created 9 sub-training sets which were subsequently used to build another set of logistic regression models.

The top 3 models by highest AUROC were selected and the common columns between the associated sub-training sets were used to obtain the best model. The test set from the first step was used against this model to obtain the final prediction.

This process was repeated for each of the simulated datasets. Using the confusion matrices from the resulting models, we computed the accuracy, precision, recall, F1-score, and AUROC scores as Kang and Zhang [2] have done for each of the final models. Then, we performed a comparison of these metrics to draw insight and suggestions for improvement on KCV-SMOTE and KFS performance with different ratios of class imbalance and SNR.

3 Results

In this section, we summarize the results of our simulation study. The first part of our simulation investigated the KCV-SMOTE and KFS method on varying proportions of fraudulent transactions, as intended by Kang and Zhang [2]. As shown in 5, the method performs very well for each of the simulated datasets indicating that it is able to handle data high degrees of unbalanced classes.



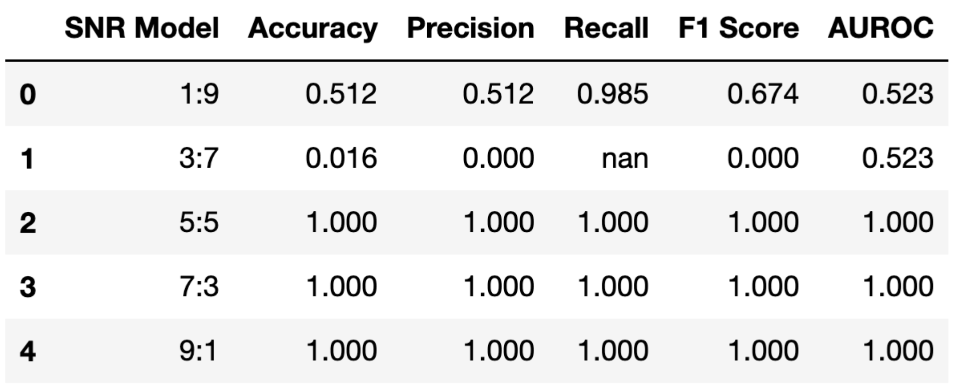
	Unbalanced Class Model	Accuracy	Precision	Recall	F1 Score	AUROC
0	0.35%	1.0	1.0	1.0	1.0	1.0
1	0.50%	1.0	1.0	1.0	1.0	1.0
2	0.80%	1.0	1.0	1.0	1.0	1.0
3	1.0%	1.0	1.0	1.0	1.0	1.0
4	1.5%	1.0	1.0	1.0	1.0	1.0

Figure 5: Unbalanced classes analysis results

The second part of our simulation study analyzed the method on different levels of SNRs in terms of predictor variables. The first group of simulated datasets had a 1% fraudulent transaction proportion. Figure 6 summarizes the results of applying KCV-SMOTE and KFS to these datasets. As shown in the first two rows, when there are more noise predictors compared to signal predictors, the accuracy, precision, and AUROC of the model is poor at approximately 51%-52%. For the case where the SNR is 1:9, recall is high at 98%. However, when the SNR is 3:7, we obtain an invalid recall result. In other words, the model did not correctly predict fraud cases nor did it falsely predict any genuine cases, making the computation for recall invalid. Overall, this tells us that the KCV-SMOTE and KFS method has unstable and poor performance when there are more noise predictors compared to signal predictors.

If we look at the last three rows in Figure 6, we can see a large difference in performance once the SNR ratio becomes even or when there are more signal predictors compared to noise predictors. This suggests that the KCV-SMOTE performs well when there is at least an even ratio of signal predictors to noise predictors.

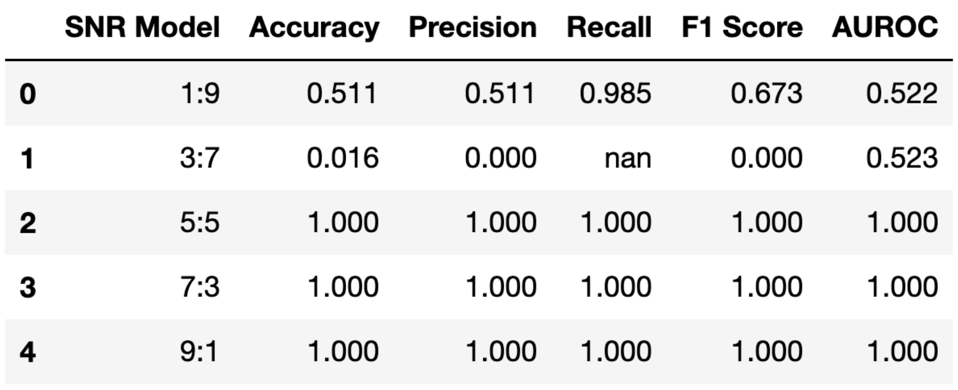
The second group of simulated datasets had a 0.80% fraudulent transaction proportion. As summarized in Figure 7, we see a very similar pattern of results for the second



	SNR Model	Accuracy	Precision	Recall	F1 Score	AUROC
0	1:9	0.512	0.512	0.985	0.674	0.523
1	3:7	0.016	0.000	nan	0.000	0.523
2	5:5	1.000	1.000	1.000	1.000	1.000
3	7:3	1.000	1.000	1.000	1.000	1.000
4	9:1	1.000	1.000	1.000	1.000	1.000

Figure 6: SNR analysis results for data containing 1% fraud proportion

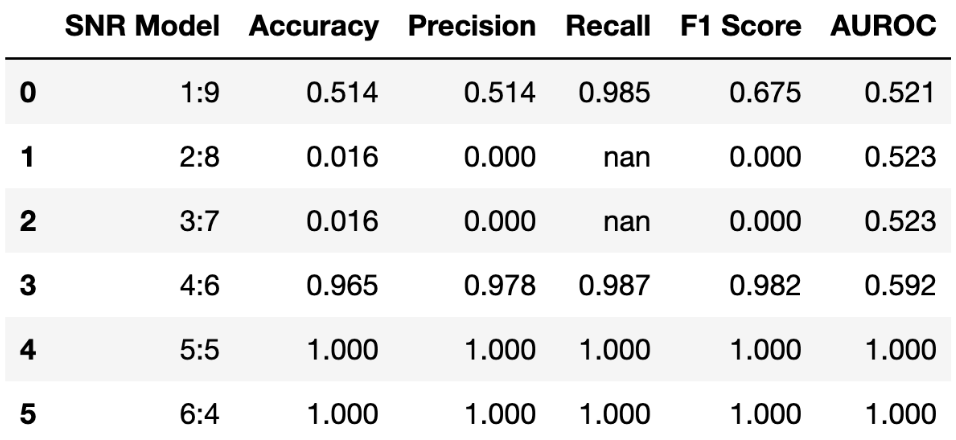
group of datasets compared to the first group. All of the metrics are unstable and poor when the SNR is low, but are high when the SNR is even or higher.



	SNR Model	Accuracy	Precision	Recall	F1 Score	AUROC
0	1:9	0.511	0.511	0.985	0.673	0.522
1	3:7	0.016	0.000	nan	0.000	0.523
2	5:5	1.000	1.000	1.000	1.000	1.000
3	7:3	1.000	1.000	1.000	1.000	1.000
4	9:1	1.000	1.000	1.000	1.000	1.000

Figure 7: SNR analysis results for data containing 0.80% fraud proportion

Likewise can be deduced from the third group of simulated datasets, which had a fraudulent transaction proportion of 0.35%. Results are shown in Figure 8 below. In this comparison, we investigated more cases where there is a higher number of noise predictors than signal predictors. We can see that the model also performs well when there are 4 signal predictors and 6 noise predictors. Accuracy, precision, recall, and F1 scores are high at 96% or greater while the AUROC is moderate at 59%.



	SNR Model	Accuracy	Precision	Recall	F1 Score	AUROC
0	1:9	0.514	0.514	0.985	0.675	0.521
1	2:8	0.016	0.000	nan	0.000	0.523
2	3:7	0.016	0.000	nan	0.000	0.523
3	4:6	0.965	0.978	0.987	0.982	0.592
4	5:5	1.000	1.000	1.000	1.000	1.000
5	6:4	1.000	1.000	1.000	1.000	1.000

Figure 8: SNR analysis results for data containing 0.35% fraud proportion

We confirmed these results once more with the fourth group of simulated datasets, which had a fraud proportion of 1.50%. Results have been omitted for the purpose of clarity.

Overall, the results of our study demonstrate two key findings. First, having an SNR of at least 4:6 signal-to-noise predictors will yield promising results using KCV-SMOTE and KFS. Second, changes in fraud proportion do not have a major impact on model performance outcomes even with varying degrees of SNRs.

4 Discussion

In this section, we discuss a number of ways we could improve the results of our work for future studies. In addition, we provide direction on how the KCV-SMOTE and KFS method can be modified to handle lower levels of SNR.

We begin by suggesting a few ways to improve the results of our work for future studies. First, experiment on different values of k parts when implementing KCV. In our study, we used $k = 5$ parts as this value is generally used when implementing KCV. However, exploring different values of k may help improve results in the study.

Second, use a more rigorous approach to select key features when performing KFS. In our study, we have chosen to include no key features. However, having expertise from the credit card industry combined with the use of a classifier such as logistic regression can better inform our choice of key features. Incorporating these key features into the KFS process may help yield better results.

Third, study more combinations of class imbalance and SNRs. In our simulation study, we only included a total of 10 predictor variables for each dataset. This gives us limited capability to explore more granular ratios of SNR. For instance, if we had 30 variables to work with, we could investigate how the KCV-SMOTE and KFS method performs under a SNR of 1:29. This would provide us a greater understanding of what the SNR threshold would be for the KCV-SMOTE and KFS method to produce effective prediction results.

Lastly, perform a similar study using real data instead. One convenient option is to use the public data set on credit card fraud collected by the ULB and Worldline [5]. Since our simulation study found that KCV-SMOTE and KFS performs well on several different percentages of class imbalance, a low percentage of fraud in comparison to genuine transactions in a real dataset would not be major concern to the performance of the method. Moreover, Kang and Zhang [2] have shown this since they have obtained a strong predictive accuracy using KCV-SMOTE and KFS in their study. To perform a similar study on a real dataset, we can treat the class imbalance proportion as fixed, assume all variables in the dataset are signal predictors, and add simulated noise predictors into the study. Since there is a greater risk involved when assuming real data as signal predictors, one can optionally perform a preliminary analysis to decide which predictors are most likely signals as opposed to noise.

We now provide an idea on how the KCV-SMOTE and KFS method can be modified to better handle lower levels of SNR. Specifically, we can introduce a feature selection step into the KCV-SMOTE and KFS process that filters out predictors that have a high chance of being noise. This step can be inserted after logistic regression is performed on each of the sub-training sets produced by KFS. Backward stepwise elimination can then be applied to reduce the number of potential noise predictors for each model, and as a result reduce the SNR in the final model. An illustration of this modification is available in Figure 9 below.

Since there is a general debate on whether stepwise elimination should be applied on linear regression model fitting, one way we could reduce the risk of potentially excluding important explanatory variables is to set a high threshold on what what predictors are considered significant and only remove the predictors that fall outside of this range. Further work is required to test this proposed modification and its ability to improve model performance for datasets with low SNRs.

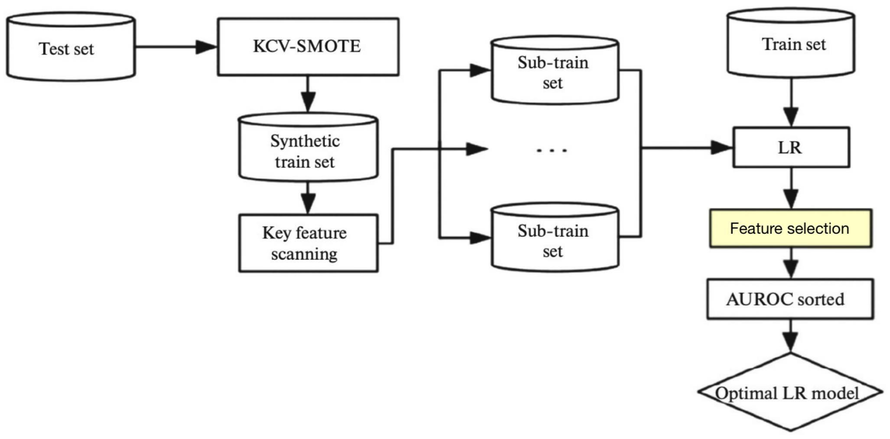


Figure 9: Proposed modification to the KCV-SMOTE and KFS method using feature selection to improve performance on low SNRs

5 Conclusion

Overall, our simulation study has provided greater insight into the effectiveness of Kang and Zhang's method [2] under different proportions of unbalanced data and ratios of signal-to-noise. The method performed well on all tested percentages of highly imbalanced data, as claimed by Kang and Zhang [2]. We also found that having at least a 4:6 SNR of predictors will yield accurate results using KCV-SMOTE and KFS while any SNRs lower than that will have poor and unstable performance. We have proposed a modification to the method that incorporates backward stepwise elimination to reduce the effects of low SNRs. Further research is required to test this approach and improve how the KCV-SMOTE and KFS process handles datasets with low SNRs.

References

[1] Fayaz Itoo, Meenakshi, and Satwinder Singh. Comparison and analysis of logistic regression, naïve bayes and knn machine learning algorithms for credit card fraud detection. *International Journal of Information Technology*, 13:1503–1511, 2021.

[2] Haiyan Kang and Hao Zhang. A new improved method for online credit anti-fraud. *Automatic Control and Computer Sciences*, 56:347–355, 2022.

[3] Yann-Aël Le Borgne, Wissam Siblini, Bertrand Lebichot, and Gianluca Bontempi. *Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook.* Université Libre de Bruxelles, 2022.

[4] Rejwan Bin Sulaiman, Vitaly Schetinin, and Paul Sant. Review of machine learning approach on credit card fraud detection. *Human Centric Intelligent Systems*, 2:55–68, 2022.

[5] Worldline and Machine Learning Group - ULB. Credit card fraud detection, 2017.