A Gamut of Locality Approaches for Applying SMOTE+ENN to Big Data

TB03 - Techniques to deal with Imbalanced Big Data

SJJHKKDF
School of Computer Science
University of Nottingham
{psxdf1,psyjjh,psxkk6,psxsj2}@nottingham.ac.uk

I. Introduction and Background

Unauthorised or fraudulent card payments impose a significant cost on the victims every year, with £1.6 billion worth of illicit transactions being blocked by the UK finance sector in 2020 alone, representing roughly 70% of total of all fraudulent transactions [1]. The incidence is only predicted to rise with increased consumer adoption of online banking and e-commerce, with credit cards accounting for a large proportion of this volume. The cost of insurance/ compensation and manual investigation into these cases is significant to the institutions providing credit services, and this also spills over into financial crime investigations and involvement of policing resources. As such the improvement of automated systems monitoring for illicit transactions to reduce these costs is acute.

Transaction data has several characteristics that make screening for fraudulent transactions challenging. Firstly the scale of the data has been growing annually at a consistent rate and is now in the order of 1.7 billion debit and credit card transactions monthly, and secondly the profile of these transactions tends to be skewed heavily in favour of the non-fraudulent transactions as despite fraudulent transactions being extremely numerous, the volume of legitimate transactions is much greater.

The data-set used here consists of 284,807 European credit card transaction instances of which 492 (0.172%) are fraudulent (denoted by the class label '1'). 2 of the 30 predictors are the output of PCA transformation applied in part to anonymize the data set. The remaining 2 are the 'Time' and 'Amount' features.

Typically there are two approaches to handling imbalanced data-sets: under-sampling the more numerous 'majority class', or oversampling (increasing the number of) the corresponding 'minority class'. The goal of this is to gain a more balanced class ratio, which should lead to more accurate training of classifiers due a reduced bias for any single class. For both of these approaches there are a variety of random & synthetic approaches available.

In the big data context it may be considered desirable to under sample due to the cost associated with increasing the volume of the data when balancing the ratios of the classes. However in the case of extremely unbalanced data the amount of information lost by under-sampling the majority class to achieve balance would be significant, hybrid approaches may find a more appropriate trade-off in this situation.

Synthetic Minority Oversampling Technique (SMOTE) is a widely used oversampling method based on the k-nearest neighbours algorithm which synthesises minority class instances at points between proximal examples of the same class, acting on the assumption that similar examples of the class will tend to co-locate in the feature space [2]. In cases where there is considerable overlap between the classes this may result instances that may be regarded as ambiguous by classification algorithms [3].

Hybrid over/under-sampling approaches have been developed to attempt to address this issue such as Tomek links or its extension ENN.

Both Tomek links and ENN (Edited Nearest Neighbours) have their basis in the k-nearest neighbour (KNN) algorithm. Instances are identified as 'Tomek links' if its nearest neighbour is of the opposing class [4]. When identified as such, either each instance in the link will be removed from the sample or else, only the majority instance. ENN is conceptually similar, removing all instances when an instance is misclassified compared to its 3 nearest neighbours. Application of ENN can be iterated until convergence to increase the volume of data removed.

Both of these "cleaning" approaches are aimed at making the decision boundaries between classes more distinct for classifiers, and are based on the observation that instances identified by these methods are likely to occur for boundary instances (between two classes) or with noisy instances. This is particularly useful when applied after SMOTE which is prone to generate noisy data where points have been interpolated between outliers [5].

KNN-based techniques are computationally expensive if conducted in a brute force manner, whereby the similarity of all points in a sample have to be calculated for each instance of interest, and this scales proportionally with the volume of data, precluding its practical usage for big data analysis. To combat this, implementations of KNN have been proposed that leverage dimensional structuring approaches such as k-d trees to enable spatially coherent splitting of the data set limit the search space used and reduce the time complexity of the algorithm and avoid the necessity of holding the whole data set in RAM. [6].

We characterise our approach by comparing the effectiveness of the application of ENN in terms of running time and performance of the Random Forest classifier.

II. Proposed Methodology

Three approaches were created to facilitate exploration and enable comparison:

1) Approach 1 - Local: Here we leverage an existing MLib implementation of the k-means algorithm ('k-means ||') which has been optimised for cluster based workflows by utilizing hybrid spill trees. Using this, instances are labelled with a cluster identifier and subsequently are aggregated into partitions by this ID, with a single node per cluster (figure 1).

We then utilise the mapPartition function to apply the imblearn library's 'SMOTEENN' function locally on each worker node/executor depending on the cluster ratio of minority to majority classes. Clustering groups instances on the basis of similarity, with more similar instances being more likely to inhabit the same cluster. By applying clustering to the dataset we hope that KNN based functions can be applied more efficiently to these feature space 'neighbourhoods' as it reduces the number of neighbours that need be calculated for each point, but does so in a manner that hopefully retains the most relevant instances, in comparison to random partitioning which will incorporate points from an even distribution of the feature space. The ability to increase the number of clusters generated means that more nodes in the cluster can be utilised which should be able to enhance the speed of these approaches. There are several costs and caveats associated with this approach:

- A large amount of data shuffling occurs at the outset, as instances are co-located into partitions based on cluster ID. We rationalise this as analogous in some sense to the indexing design pattern, in which computational effort is expended up-front to make future queries more efficient, if an effective cluster configuration is found for a data-set it can be preserved for future queries using a format like parquet.
- In relation to this, the cluster boundaries may be generated in a manner that artificially separates instances, which could cause issues, for example by in the case of SMOTE, if distal minority instances are grouped, noisy synthetic instances may be more generated.
- Very large data sets may preclude the use of k-means
- Too many clusters may overly fragment neighbourhoods of instances and prevent effective application of SMOTE.
- It requires more worker nodes than clusters.
- Clustering is fundamentally a method for grouping based on similarity, if the similarity that is captured represents the classes then the resultant clusters may be too homogeneous to support SMOTE. Our hope is that the clusters are of a moderate granularity for the dataset so that it provides an effective spacial organising method, but still allows class balancing to be effected.
- Using a high number of clusters for highly imbalanced data may result in numerous clusters that do not have the presence of minority instances. If the multiplication factor for SMOTE is based on the intra-partition numbers of the classes, then this will lead to a low global total number of synthetic instances. This could be mitigated with the use of broadcast variables that establish a baseline multiplier for min instances.

Algorithm 1 Approach 1:

```
$df \leftarrow loadFile(dataFile).cache()$
$df\_features \leftarrow df.vectorise(*Features)$
$df\_cl \leftarrow df.kmeans(n, df.Features)$
$df\_clpart \leftarrow df\_cl.partitionBy(k, df\_cl[ClusterNo])$
$output \leftarrow df\_clpart.mapPartition(SMOTEENN)$
```

With consideration given to the above shortcomings, it is hoped that this approach makes the performance promises that can result from the classical divide and conquer paradigm.

With consideration of the above we wanted to investigate how the performance compared against existing 'global' implementations of SMOTE+ENN that use alternative methods to reduce the computational burden to only the most relevant features.

2) Approach 2 - Global: We compare against a 'global' approach that uses an existing global SMOTE implementation [7] that leverages the Locality Sensitive Hashing (LSH) functionality provided in Spark ("BucketedRandomProjectionLSH") for finding the kNN [8] . LSH represents an alternative mechanic to create spacial coherency, whereby a similarity hash is applied to each instance in the set to generate a reduced subset of comparison candidate instances for each instance (Figure 2).
We used this same mechanism to implement a complementary 'global' ENN function. No clustering was performed for this method.

Algorithm 2 Approach 2:

```
$df \leftarrow loadFile(dataFile).cache()$
$df\_features \leftarrow df.vectorise(*Features)$
$df.globalSmote()$
$df.globalENN()$
```

3) Approach 3 - Hybrid: Approach 3 hybridises the above approaches. Clusters are calculated across the instances as

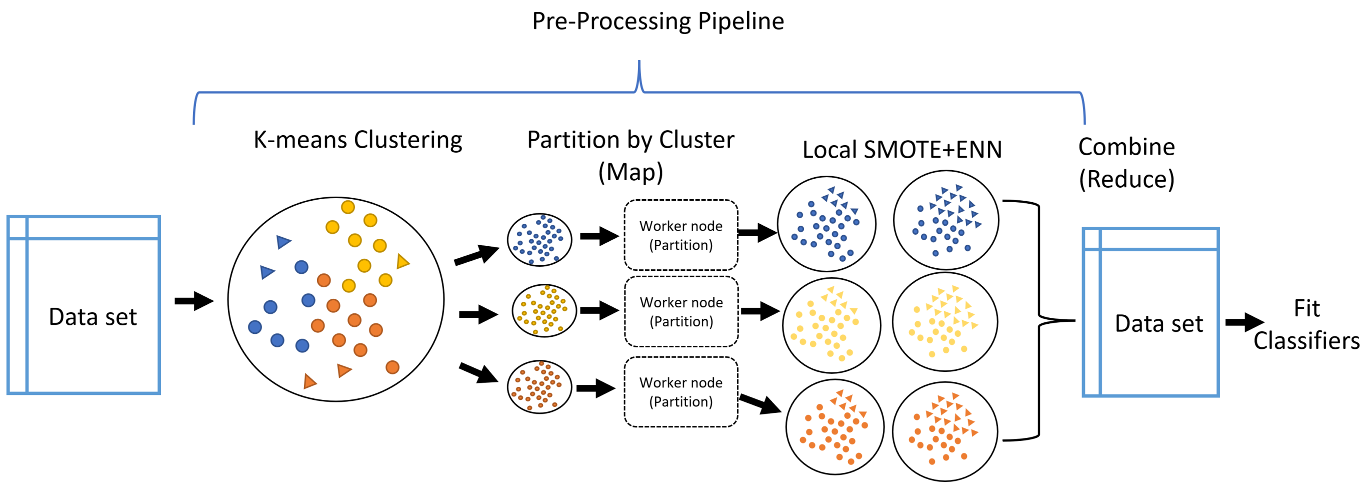

Fig. 1. Approach 1: Local SMOTE+ENN

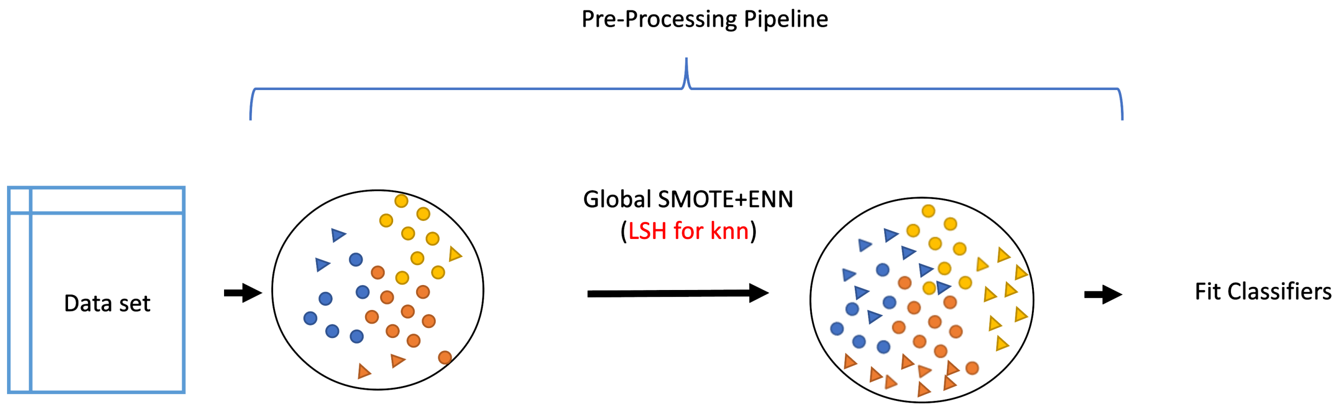

Fig. 2. Approach 2: Global SMOTE+ENN

described in approach 1. The clusters are assigned to separate data frames, and the approximated 'global' algorithms are applied to each depending on the ratio of minority to majority classes of each cluster. By applying LSH within identified clusters it is hoped that the similarity measures will be more accurate and therefore improve the performance. A weakness of this approach is that Spark operates on separate data-frames sequentially which under-utilises cluster resources and is highly inefficient. It was beyond the scope of this paper to deploy each clustered data frame to a separate spark instance.

Algorithm 3 Approach 3

```
  $df \leftarrow loadFile(dataFile).cache()$
  $df\_features \leftarrow df.vectorise(*Features)$
  $df\_cl \leftarrow df.kmeans(n, df.Features)$
  **for** $df\_cl$ in $df\_clusters$ **do**
      **if** $df\_cl.classRatio$ between $0$ and $1$ **then**
          $df\_cl$.globalSmote(Mult_factor);
          $df\_cl$.globalENN();
      **end if**
  **end for**
$df \leftarrow union$(df_cl)
```

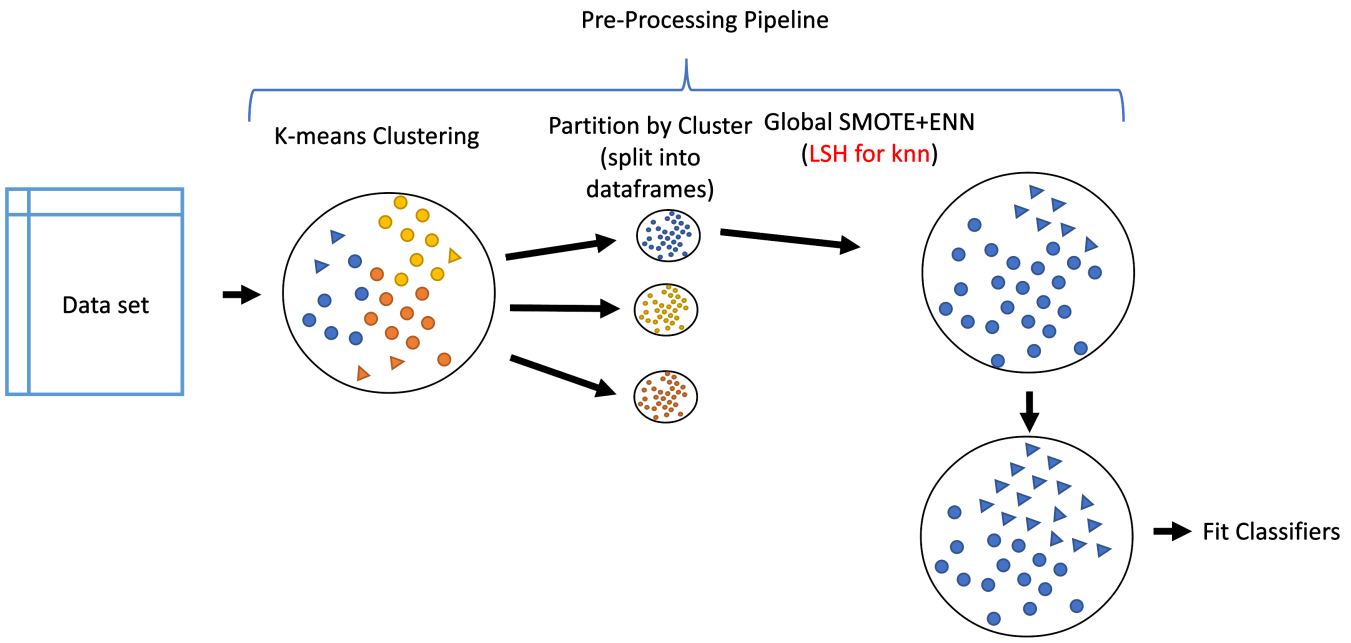
Pre-Processing Pipeline
Data set
K-means Clustering
Partition by Cluster (split into dataframes)
Global SMOTE+ENN (LSH for knn)
Fit Classifiers

Fig. 3. Approach 3: Hybrid SMOTE+ENN

III. Experimental Set-up

This paper focuses on the performance of different implementations of the SMOTE+ENN pre-processing approaches as applied to big data. As such we are predominantly interested in the running time and output balance ratio, with secondary consideration given to the resultant classification performance. SMOTE implementations were parameterised with the same multiplication factor across the hybrid and global approaches, with the local approach using an alternative configuration.

The training/testing split utilised for generating classifier metrics occurred after and independently of the pre-processing stages.

For the 30 and 50% test cases for each approach the split was generated to conserve the presence of the minority class. This was done as the proportion of minority class instances is very small and therefore random variation could have unduly impacted SMOTE.

Collected Metrics:

- Pre-processing Running Time
- Resultant Class Ratio
- Area Under ROC Curve
- Area Under Precision Recall Curve
- Accuracy
- F1
- Class Ratio

The pre-processing approaches were applied to limited subsets of the whole data set at the 30%, 50% and 100% level to provide an indication of scaling performance, and metrics recorded for each of these.

The resulting balanced data sets were fitted to a random-forest classifier as a validation mechanism.

IV. Results and Discussion

This section presents the metrics observed from the discussed approaches. We analyse each approach individually and compare between them.

Clustering was utilised for two of the approaches (Local & Hybrid). For the calculation of the comparison metrics a clustering value of 6 was used. It was timed separately and on average took 8-10 seconds. Clustering time is included in the overall pre-processing runtime recorded in the metrics tables.

1) Local:

- Key classifier performance metrics such as AUC, Test Area, Accuracy and F1 remained broadly consistent across the 30, 50, and 100% groupings.
- The resulting class balances did not change as the overall size of the data increased due to the sampling parameterisation used in the SMOTE function.
- The increase in runtime observed as the volume of data increased was significant, taking roughly 10x longer for the full dataset compared to 30% of the dataset.
- We attributed this increase to a larger average volume of instances per partition, and this prompted further investigation into increasing number of clusters for this approach.
- The results from this investigation are shown in table IV and illustrate that the performance of the classifier remained relatively constant for different numbers of clusters/partitions.
- However the runtime reduced by a factor of 10 when the number of clusters was increased from 6 to 100.

- We attribute this to more efficient parallelisation as each executor was operating on smaller sets of instances.
- The local approach did outperform the other approaches when considering the resulting class ratio, with a 50/50 split between the classes, compared to an average of 1.95% and 5.18% for approach 2 and 3 respectively.

2) Global:

- Key classifier performance metrics such as AUC, Test Area, Accuracy and F1 remained broadly consistent across the 30, 50, and 100% groupings.
- The pre-processing runtime diminished slightly with increased dataset size.

3) Hybrid:

- As the size of the data set increased the running time reduced, which was the inverse of our expectation. A proposed mechanism for this is for the granularity of the similarity hashing employed is reduced in the instance subsets represented by the clusters, thereby resulting in fewer comparison candidates being generated during the KNN-based SMOTE+ENN, but this is conjecture and area for further study.
- The oversampling proportions increased as the overall size of the data decreased, this was likely due to the sampling parameterisation used in the SMOTE function.
- Key classifier performance metrics such as the Test Area Under ROC, Test Area Under PR, Accuracy and F1 remained broadly consistent across the 30, 50, and 100% groupings.

4) Comparison: Taking into account the above, we can state that:

- No method induced a materially detrimental effect on the performance of the classifier.
- The local approach took significantly longer than the other two approaches. One factor that may have a bearing on this is that the implementation of SMOTE applied to the local approach was parameterised inconsistently from the other two approaches, which may have resulted in much higher levels of synthesis of the minority class.
- This discrepancy in balance ratios did not have a material impact on classifier performance.
- The local approach demonstrated a marked increase in area under the precision recall curve.

One observation that affected this study is that the features of the data in question clearly provided extremely clear signals for the classifiers to train on. As such the the classification metrics were consistently very good, and this may have masked any observable effect that the pre-processing approaches were having on the classification performance.

% of Data	Test Area Under ROC	Test Area Under PR	Accuracy	F1	Time Taken	Class Ratio %
30	0.9584	0.8085	0.9950	0.9955	412	$\sim$100
50	0.9830	0.8230	0.9949	0.9959	1147	$\sim$100
100	0.9788	0.7199	0.9965	0.9973	3970	$\sim$100

TABLE I
PERFORMANCE OF THE LOCAL APPROACH

% of Data	Test Area Under ROC	Test Area Under PR	Accuracy	F1	Time Taken	Class Ratio %
30	0.9449	0.7204	0.9991	0.9991	83	1.95
50	0.9494	0.7367	0.9990	0.9990	72	1.94
100	0.9617	0.7616	0.9992	0.9993	73	1.96

TABLE II
PERFORMANCE OF THE GLOBAL APPROACH

% of Data	Test Area Under ROC	Test Area Under PR	Accuracy	F1	Time Taken	Class Ratio %
30	0.9469	0.7768	0.9991	0.9992	916	10.37
50	0.9390	0.7496	0.9992	0.9992	745	5.80
100	0.9628	0.7296	0.9993	0.9993	570	2.57

TABLE III
PERFORMANCE OF THE HYBRID APPROACH

TABLE IV
LOCAL APPROACH RUNTIME VS CLUSTER NUMBER

No. of Clusters	**6**	**40**	**65**	**100**
Area Under ROC	0.979	0.971	0.969	0.971
Area Under PR	0.719	0.719	0.730	0.726
Accuracy	0.997	0.991	0.991	0.993
Precision	0.999	0.999	0.991	0.993
Recall	0.996	0.993	0.991	0.993
F1	0.996	0.992	0.994	0.995
Total Runtime (s)	**3970**	**1013**	**641**	**371**

V. CONCLUSIONS

Scarcity of minority classes in big data is an uncommon problem, but can occur in impactful contexts, particularly in the example of clinical data [5]. We have demonstrated three prospective approaches to pre-processing large data sets in practical time-frames, that can attempt to address the class imbalance by improving the running time of the relevant SMOTE+ENN oversampling techniques, with the aim of improving or enabling classifier performance. The focus of our study was to implement a divide and conquer based implementation that leveraged clustering as a more intelligent division measure. We identified several challenges with this approach that likely indicate that it is not suitable for most data sets, particularly as there are global similarity searching approaches that are well explored and available implementations are extremely efficient [9]. Nevertheless we feel that it is worth further investigation to investigate whether this approach can be refined.

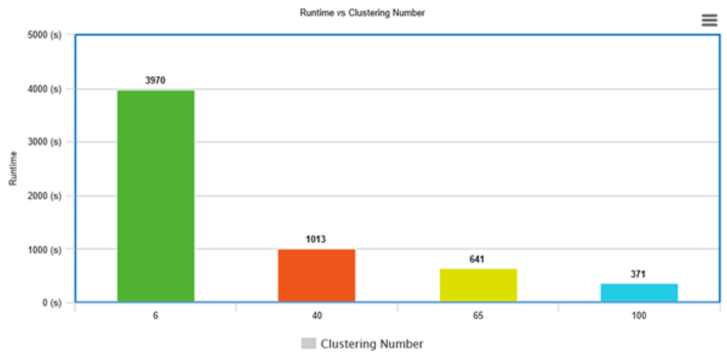

Fig. 4. Local Approach Performance

References

[1] UK Finance Report, "FRAUD - THE FACTS 2021."

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique." [Online]. Available: https://arxiv.org/abs/1106.1813

[3] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," vol. 6, no. 1, pp. 20–29. [Online]. Available: https://dl.acm.org/doi/10.1145/1007730.1007735

[4] A. More, "Survey of resampling techniques for improving classification performance in unbalanced datasets." [Online]. Available: http://arxiv.org/abs/1608.06048

[5] G. E. A. P. A. Batista and M. C. Monard, "An analysis of four missing data treatment methods for supervised learning," vol. 17, no. 5, pp. 519–533. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/713827181

[6] M. M. A. Patwary, N. R. Satish, N. Sundaram, J. Liu, P. Sadowski, E. Racah, S. Byna, C. Tull, W. Bhimji, Prabhat, and P. Dubey, "PANDA: Extreme scale parallel k-nearest neighbor on distributed architectures." [Online]. Available: https://arxiv.org/abs/1607.08220

[7] hwang018, "smote_spark.py." [Online]. Available: https://gist.github.com/hwang018/420e288021e9bdacd133076600a9ea8c

[8] "Approximate nearest neighbor search." [Online]. Available: https://spark.apache.org/docs/2.2.3/ml-features.htmlapproximate-nearest-neighbor-search

[9] T. Christiani, "Fast locality-sensitive hashing frameworks for approximate near neighbor search." [Online]. Available: https://arxiv.org/abs/1708.07586