Similarity-based One-Class Classification for Fraud Detection in the Big Data Context

Tze Ee Yong*, Jiachenn Choong†, Hui Shen Sam‡, Dong Jun Lee§
School of Computer Science,
University of Nottingham
Nottingham NG7 2RD
Email: *hfyty1@nottingham.ac.uk, †hfyjc3@nottingham.ac.uk, ‡hcyhs1@nottingham.ac.uk, §hcydl1@nottingham.ac.uk

***Abstract*—Fraud detection is a popular problem to solve in data analytics as solutions for it bring about positive impact. However, solving fraud detection problems is non-trivial as we are often dealing with severely imbalanced datasets as fraudulent transactions do not happen regularly. Additionally, this problem becomes harder as the number of transactions scale up due to inefficiencies in the solutions for fraud detection, particularly for similarity-based solutions as the distance to each instance in a model needs to be recalculated for every incoming instance. To see whether such solutions are still relevant for processing big data, we developed two such approaches (based on k-NN and k-Means clustering) and evaluated them in the big data context. To improve its scalability, we modified the approach for one-class classification, which reduces the model size when predicting new instances. Our results show that our k-Means approach has decent scalability, while our k-NN approach achieved a better prediction performance in the anti-learning sense.**

Keywords*—*One-class classification, Fraud Detection, k-Nearest Neighbours, k-Means clustering, big data

I. Introduction

Fraud detection has been one of the main topics in banking, finances, and insurance companies. In 2020, the banking industry had protected customers with over £1.6 billion of fraud stopped. With the increase in online transactions through the usage of debit/credit cards, the number of fraudulent transactions also increases. Therefore, there is a need to design fraud detection algorithms for preventing loss in these companies. A fraudulent transaction is the illegal usage of debit/credit card details without the knowledge of the real cardholder. Fraudulent transactions usually show an abnormal expenditure pattern compared to legitimate transactions. The main task in fraud detection is to learn the patterns of fraudulent transactions and then classify whether a new transaction is fraudulent or non-fraudulent. However, fraudulent transactions occur rarely and very few in number compared to the high volume of legitimate transactions. In this case, the dataset is heavily imbalanced where we have a large number of instances for a class (i.e. non-fraudulent transactions), and very few instances for the other class (i.e. fraudulent transactions). Classification algorithms can be used to identify whether a transaction is fraudulent or not. It can be performed either using binary classification or one-class classification. To deal with the problem where the dataset is heavily skewed towards the negative class (i.e. non-fraudulent transactions), we have chosen a one-class classification approach.

We will look at two popular algorithms that classify an instance based on a similarity measure, which are k-Nearest Neighbour (k-NN) and k-Means clustering. The k-NN algorithm classifies a test instance by first calculating the similarity of that instance with each instance in the training set. Then, the test instance is assigned a class based on the majority vote of its k nearest neighbours. This means that the k-NN algorithm has to store each training instance in memory, and then calculate the similarity of the test instance with each training instance to make a prediction. For k-Means clustering, an instance is classified based on its similarity to the cluster centers found in the training set. It is an iterative process to find the optimal cluster centers in the training set. For both algorithms, the computation is expensive as they need to scan through each instance in the training set in order to classify an instance. The inefficiency of both algorithms becomes more noticeable when the training set is large. Due to the way of how both the algorithms work, they do not scale well with large datasets.

The main goal of our work is to test the scalability of the k-NN algorithm and k-Means clustering algorithm for large datasets. Thus, we implemented our solution in the Big Data framework. Big data can be defined by 5Vs, which are volume, velocity, variety, veracity, and value. A fraud detection system has to deal with a huge amount of data as transactions are occurring almost at all times. The huge amount of transactions refers to volume, and the number of transactions occurring frequently refers to velocity [1]. The Big Data framework enables the processing of data in a parallel and distributed fashion, which improves the scalability of the k-NN algorithm and k-Means clustering algorithm as compared to using a non-Big Data approach that only utilises a single CPU. In our study, we implemented these two similarity-based algorithms in a one-class classification approach to deal with the problem of an imbalanced dataset, and attempt to improve their scalability in the context of Big Data.

The remaining of the paper is organised as follows: Section II summarises the related works used to deal with imbalanced dataset and techniques used in fraud detection; Section III describes in detail the methodology we used to implement both one-class k-NN and k-Means algorithms to classify fraudulent and non-fraudulent transactions; Section IV discusses the results of our solution when predicting on the test dataset, and

its efficiency in terms of total processing time when handling datasets of different sizes; The conclusion is drawn in Section V.

II. Literature Review

This section is split into two parts, the first being a general overview of approaches on tackling imbalanced datasets followed by a brief summary of related literature in respect to works related to fraud detection with some work pertaining to Big Data approaches.

A. Handling Imbalanced Datasets

1) Feature Selection and Extraction Techniques: Feature extraction techniques have been utilised in various research domains that deal with data imbalance such as the medical domain by [2] which combines Principal Component Analysis (PCA) to find the optimal subset of features and SMOTE to pad the ratio of minority-majority class samples to obtain a more balanced class distribution.

2) One-Class Classification: The main objective of one-class learning is to more accurately delineate and distinguish decision boundaries for a given learnt single class in a binary classification problem while considering the data points that do not fall under the decision boundaries as anomalies or outliers. One-class classification has proven to be effective when utilised in critical domains where data usually suffers from a class imbalance in which the number of normal examples far outweigh the number of abnormal examples with the goal of accurately pinpointing abnormal behaviour. The researchers in [3] utilise one-class classification in a medical imaging context by combining one-class classification and a deep-learning approach. In their approach, they use an unsupervised deep learning model to learn and characterize the normal classes followed by classifying test images by patches, assigning a class label which are normal or otherwise (abnormal) in the patch. In [4], by using one-class classification, bots can be pinpointed based off their deviation of behaviour compared to legitimate human accounts due to the training on "normal" human behaviour on Twitter. The researchers show that their approach outperforms the use of binary supervised classifiers,due to the difficulty of compiling a representative set of anomalous classes for the bots as it requires extensive justification and verification to prove that bots exhibit certain behaviours.

Another one-class classification approach by [5] proposes a weighted one-class classification technique in conjunction with k-NN to incorporate background knowledge of the minority class examples to the classifier during training. The embedding of background knowledge allows the fine-tuning of the decision boundary formation by weighting the minority class examples. The category of a minority label data point is determined by a k-Nearest Neighbours algorithm, based on the number of neighbours matched. Their strategy achieved good performance and outperformed various classifiers in comparison. However, it should be noted that their approach is hugely memory intensive due to the need of storing all k-Nearest Neighbours in memory, thus posing a challenge when it is required to be up-scaled.

3) k-Nearest Neighbours (k-NN): Despite the huge memory overhead of using k-NN which may hinder its performance in Big Data context, there exists an implementation of k-NN for classification tasks [6] adjusted to suit Big Data approaches. The paper states that the classification approach is broken down into two steps, the first step is training by clustering data points using a technique named Landmark-based Spectral Clustering (LSC) [7]. The technique has low space complexity due to compressing the original samples by selecting random representative samples of the dataset using k-Means. The testing process then uses k-NN on the clusters created from k-Means by the LSC algorithm to classify new instances that enter the cluster. The research achieved satisfactory time performance on popular datasets such as MNIST, Gisette and proves that k-NN is scalable to a Big Data context, provided some pre-processing steps.

4) k-Means: K-Means has also been applied as a big data approach in the domain of fault classification in [8] when dealing with imbalanced datasets. The authors introduce an improved k-Means algorithm which incorporates naive Bayes as a classifier following k-Means. The majority class is first clustered into k sub-classes and the size of the sub-classes is determined by a threshold to prevent an imbalance in quantity of data in the sub-classes. Next, using a naive Bayes classifier, the decision boundary of the minority class is approximated, and by further sub-classing the majority class, the decision boundary gets closer to the correct approximation. The authors introduce a MapReduce implementation of the algorithm and results show that their implementation of k-Means Bayes algorithm achieves promising accuracy in fault classification. A drawback is that the paper does not compare thoroughly the throughput of the algorithm and therefore unable to fully validate its feasibility in Big Data.

B. Fraud detection

The domain of fraud detection is prone to the issue of imbalanced classes in datasets due to the volume of normal transactions compared to fraudulent transactions. Various approaches have been tried and tested in the field of fraud detection such a hybrid of decision trees and support vector machines (SVM) with strategic re-sampling strategies [9], Convolutional Neural Networks (CNN) to capture the intrinsic features and patterns of abnormal behaviour in data [10].

However, one of the most popular methodologies in fraud detection is the usage of one-class classification techniques due to the imbalanced classes nature of real-world datasets. One of the widely adopted strategies in this task is the One-Class SVM (OCSVM)[11] by the usage of a hypersphere to separate one-class of samples from the other class. A recent advancement in the field of one-class fraud detection refers to the work by [12], using one-class adversarial nets (OCAN) in two phases during training. The first phase encodes the activity sequence of normal users into feature representations using a

LSTM encoder, the second phase trains generative adversarial nets (GAN) to generate samples that are complementary to the representation of normal users which are potentially fraudulent users. The objective function of the discriminator is to then discriminate and separate normal users and their complementary counterparts. The advantage of OCAN are that it does not require labeled fraudulent data, thus removing the need for a manually composed training dataset, potentially enabling a more adaptive approach towards different domains.

Sk et al. [1] implemented a fraud detection system that uses the Particle Swarm Optimisation Auto-Associative Neural Network (PSOAANN). In their work, the PSOANN model was trained on non-fraudulent transactions, and then tested on the fraudulent transactions. AANN is a type of neural network where the number of input nodes is the same as the output nodes, and the model learns to predict the input variables at the output layer. In other words, AANN is non-linearly transforming the original input variables into a set of perturbed values. After the training process, the model learns the characteristics belonging to the non-fraudulent transactions. The weights are updated through the Particle Swarm Optimisation (PSO) algorithm. The classification of a transaction is achieved by computation of relative error for the features present in the input and output of the PSOAANN. If the relative error is greater than a pre-defined threshold, then it is a fraudulent transaction, otherwise it is a non-fraudulent transaction. The authors also implemented the algorithm using Spark where the execution time is improved due to data locality that the local file system enables.

In [13], the authors use k-Means as their approach on the fraud detection of a signaling dataset. The authors first utilise PCA as a pre-processing step for dimensionality reduction, taking only the first 6 principal components as it already contains a large amount of the original information. K-Means clustering is applied next and analysis on each cluster present was performed. From cluster analysis, it can be seen that certain clusters with prominent features such as low talk time ratio and low percentage of successful calls can be classified as fraud instances.

In the context of big data, the authors in [14] propose a hybrid framework highly focused on using Big Data technologies for online credit card fraud detection. The authors combine many design ideas from prominent credit card fraud detection works, which include an ensemble of algorithmic techniques to detect, using both supervised in conjunction with unsupervised learning techniques, and various filters in a multilayered hybrid framework. The multilayered framework is tailored to suit Big Data needs by optimising factors that pertain to Big Data performance, such as input-output throughput for MapReduce functionality using the Hadoop Distributed File System (HDFS), Apache Hadoop for distributed fraud detection model training and by using Spark Streaming to stream real-time data for fraud detection. The proposed architecture is able to support up to 100 million concurrent transactions, suggesting the framework is scalable to handle real-time data streams.

C. Summary of Literature

To tackle the challenge of imbalanced datasets, various techniques were utilised in literature that consisted of resampling techniques, feature extraction and selection, and classifiers such as the one-class classifier that uses the minority class as its training data. Existing literature regarding similarity-based approaches (such as k-NN), one-class classification and Big Data that were reviewed are an intersection of either a) similarity-based approaches & Big Data [6] b) one-class classification & Big Data. Therefore, after surveying the gaps in the literature, our work focuses on the combination of using a similarity-based one-class classification approach with the evaluation and discussion of its scalability towards Big Data.

III. Methodology

To determine if similarity-based approaches for one-class classification can be scaled to handle big data, we evaluated two simple algorithms, namely the k-Nearest Neighbours (k-NN) algorithm and the k-Means clustering algorithm. As these algorithms are traditionally designed for binary or multiclass classification tasks, we modified them for one-class classification. Additionally, as behaviours in fraudulent transactions tend to evolve over time, we developed an approach of simulating updates to our model with new labelled fraudulent instances, which allows our model to adapt to the new changes and evolve over time.

We tested our approach on the IEEE-CIS Fraud Detection task[1], which contains 590,540 transactions, with 20,663 fraudulent transactions (3.5%) and 569,877 non-fraudulent transactions (96.5%). As the dataset is severely imbalanced, this is a suitable dataset for us to test our one-class classification approach on. Our solutions are implemented mainly using PySpark, and is broken down into two phases: Phase 1 for building our initial model, and Phase 2 for an iterative improvement of our model over time. Fig. 1 provides an overview of the entire pipeline.

A. Our Pipeline

For our task, there are two components for the training dataset: the transactions dataset and the identity dataset. As the method we are using is a one-class classification technique, we are only interested in one class from the dataset, namely the fraudulent transactions class. However, to conduct our validation tests, approximately 500 of each fraudulent and non-fraudulent transactions are reserved. This is done as the ground truths are not provided in the test dataset.

The operations here are done using the PySpark DataFrames API as it is fast due to its parallelisation to several worker machines. In Phase 1, the datasets are loaded from the HDFS and we first merge the two datasets and filter out non-fraudulent transaction, leaving us with our initial dataset to work with. In total, this dataset comprises of approximately 20,163 fraudulent instances with 432 features each, excluding 500 of them that were reserved for validation testing. Our

[1]https://www.kaggle.com/c/ieee-fraud-detection/overview

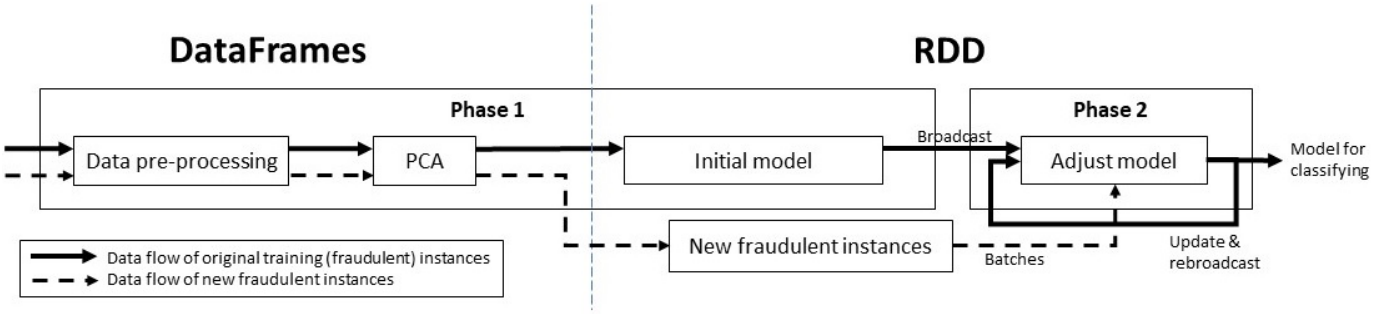

Fig. 1. Our proposed pipeline. The arrows indicate the data flow of fraudulent instances.

next step is to pre-process and clean our dataset to manually remove noise. This is done by first identifying and removing redundant features, which we found there are 2 in total (id_33 and DeviceInfo). Then, we pre-process our dataset as such:

- Reduce string features with a wide range of values (P_emaildomain, R_emaildomain, id_30, and id_31) into confined categories
- Convert TransactionDT to be the time delta relative to each day. This is done by taking its modulo 86,400 as there are 86,400 seconds per day

To simulate new fraudulent instances being passed into our model over time, a portion of the train dataset is reserved (we reserved approximately 50%), to be used in Phase 2. This leaves approximately 10,000 fraudulent instances in the train dataset for the next step.

After pre-processing our dataset, PCA is applied for dimensionality reduction and feature extraction. This produces a new dataset that is more information-rich, while having a smaller size. The number of principal component dimensions to use is determined by the fraction of its variance, where dimensions with a fraction larger than 1% is used as a smaller fraction resulted in diminishing returns. For the k-NN approach, our initial model consists of the principal components generated from PCA, which are treated as model instances in our model. For the k-Means approach, we further process the initial model by calculating the radius of each cluster centre, and removing instances that are within radius of a cluster centre. The radius is calculated as such:

$$R_i = \alpha * \frac{size_i}{N}$$

where R_i is the radius of the ith cluster, $size_i$ is the ith cluster's size, N is the total number of instances, and α is a hyperparameter to scale the radius. PCA is also applied on the reserved instances as we need to condense the reserved instances into the same space as the instances in our model. The data is then normalised with RobustScaler to handle outlier instances and StandardScaler to normalise them.

To parallelise the operations in Phase 2, our model is broadcasted to all the worker nodes. This allows each worker node to individually perform its task on its local model without having to perform as much expensive read and write operations between it and the driver in order to update our model. Although this shifts our approach to a "local" one, this has negligible difference in the model structure. However, the speed of handling incoming new fraudulent instances compared to the "global" approach is significantly improved as we are effectively batching updates to our model, instead of passing instances into it one at a time.

In Phase 2, we improve our initial model by merging existing model instances and adding new instances to our model. This allows our model to adapt to new trends or behaviours of fraudulent transactions over time.

Phase 2 involves 3 steps. In the first step, the instances that are reserved from Phase 1 are separated into batches, where the size is defined by the user. The batches are then converted into RDDs for us to perform Map and Reduce operations. This step allows us to perform batched updates to our model, as passing new instances one at a time is largely inefficient and can clog up the system if there is a sudden increase in volume of new incoming instances. In step two, for the k-NN approach, we first define a hyperparameter termed "radius" that acts as the threshold distance between the new instance and a model instance. We then perform a Map and a flatMap operation on the new instances to calculate the distance between all model instances and a new instance, and retrieve the model instances that are similar (ie. the distance is smaller than the radius) to the new batch instance. This is to reduce the model size as these model instances are effectively occupying the same local space as the new instance. This also allows us to retain a relatively accurate representation of the distribution of fraudulent instances. Next, a reduceByKey operation and an additional Map operation are performed to calculate the merged instance to be added into our model. This is done by taking the average instance of all similar instances and the new instance. By merging these instances with the new instance, we can get the average instance of this local group, which provides a general representation of the fraudulent transactions in this local space.

For k-Means clustering, we first compute the cluster centres to find the average instances that represents the distribution of fraudulent transactions. To determine the ideal number of clusters to use, we compute the silhouette score for k in the range [2, 20), and selecting the one with the highest score. Silhouette is a scoring mechanism for measuring how distinct

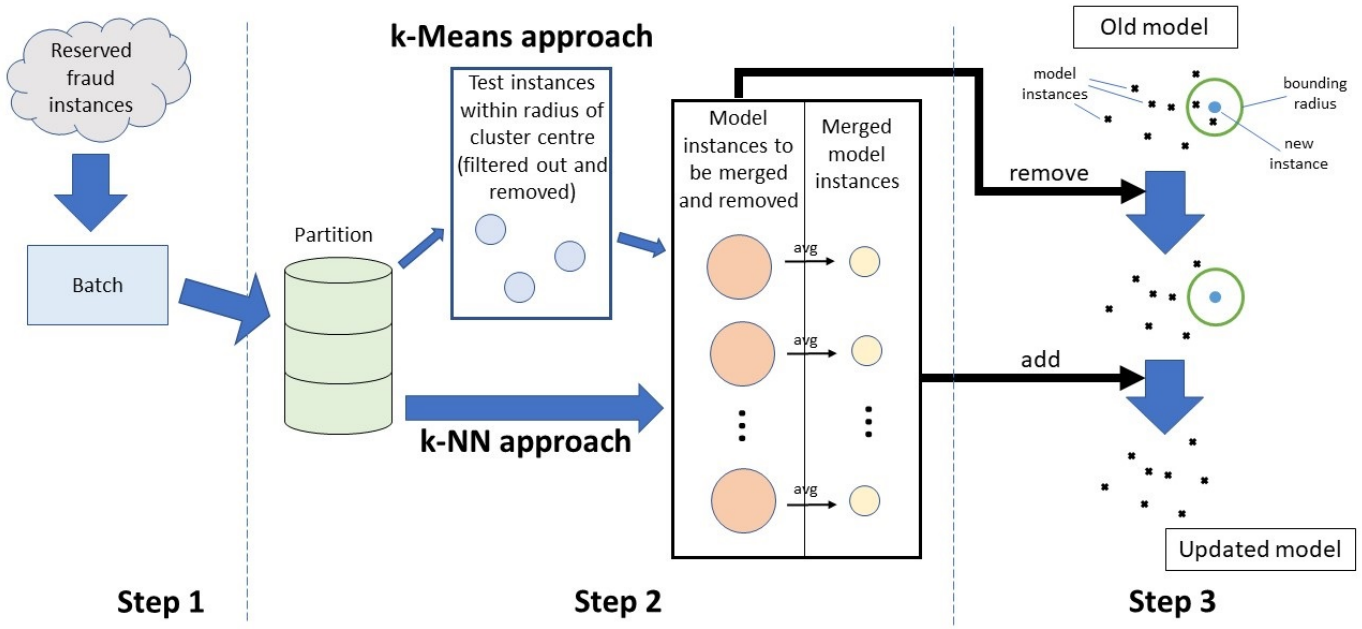

Fig. 2. Phase 2 of our proposed pipeline. This phase is split into 3 steps: 1. batching reserved new instances, 2. retrieve similar model instances and calculate merged instances, and 3. update model.

each cluster is from one another, where a higher score indicates more separation.

After this, we perform the same operations in Steps 1 and 3 as in k-NN. For Step 2, we first perform a Map and Filter operation to remove new instances that are within the radius of any cluster centre before doing the operations as described for the k-NN approach.

Once we obtained the model instances for removal and calculated the merged instances, we can update our model in step three. Although this is simply done by removing the model instances we have marked for removal and adding the merged instances into our model, this is the most expensive part of the entire pipeline. This is because it involves a lot of data movement between the driver and the worker nodes as the model is collected and updated in the driver, then rebroadcasted to the worker nodes for the next iteration. However, this is necessary for updating our model in batches as we need to aggregate all updates from the worker nodes in every iteration, and we accept that there are no workarounds for this. An overview of Phase 2 is illustrated in Fig. 2, and the algorithm for each approach is shown in Algorithm 1.

Ideally our model will continuously update as new fraudulent instances are identified and passed into our model. However, for the purpose of our testing, we stop updating our model after going through our entire reserved instances.

B. Predicting Test Instances

Once we have our model, we can predict the test instances. The test instances are obtained from the test dataset and they are first processed in the same way as the model instances and the reserved instances. This converts the test instances to the same data space as our model instances. Then, for the k-NN approach, for each test instance, we assign a radius to it and gather the model instances that are within distance of it the same way as was done in Phase 2. Next, a confidence score determining how confident the model is about the test instance being a fraudulent transaction is calculated. This is done by dividing the total number of similar model instances to the test instance by a hyperparameter k, where k is the number of model instances to prescribe the test instance as fraudulent with 100% confidence. If the number of similar model instances is larger than k, the confidence score is clipped to 100%. To further assign a class for the test instance, a threshold (hyperparameter) for the confidence score can also be set. This means that test instances with a confidence score higher than the threshold are considered as fraudulent, and instances with scores below it are non-fraudulent. The entire process is done with two Map operations. For the k-Means approach, we first classify all instances that are within the radius of any cluster centre as fraudulent. Then, a radius is assigned for each remaining test instance, and these instances are classified as fraudulent if there exists a model instance that is within distance of it, and non-fraudulent otherwise. The k-Means operation is done in four Map operations. We illustrated an example of this process in Fig. 3 along with each algorithm's description in Algorithm 2.

IV. Results and Discussion

As our goal is to test the performance of similarity-based approaches in the big data context, the focus is on the efficiency and processing time of our pipeline. However, we also evaluated the effectiveness of the approached with accuracy and AUROC.

Algorithm 1.1: Improving model with reserved instances (k-NN)

```
Input:
r = user-defined radius
reserved = reserved instances, split into batches
model = broadcasted model (contains model instances)

Output:
model' = the final updated model

Algorithm:
for batch in reserved:
    batch = sc.parallelize(batch)
    # Calculate distance between new instances and model instances
    batch = batch.map(lambda x: calcDist(x,model.value))
    # Gather model instances for removal and merging
    similar = batch.flatMap(lambda x: filterSimilarInstances(x,r))
    remove_instances = similar.collect()
    # Merge instances and update the model, then rebroadcast it
    merged = similar.reduceByKey(add).map(lambda x: x/numInstances)
    updated = model.value - remove_instances + merged.collect()
    model = sc.broadcast(updated)
model' = model
```

Algorithm 1.2: Improving model with reserved instances (k-Means)

```
Input:
r = user-defined radius
reserved = reserved instances, split into batches
centres = broadcasted cluster centres
model = broadcasted model (contains model instances)

Output:
model' = the final updated model

Algorithm:
for batch in reserved:
    batch = sc.parallelize(batch)
    # Calculate distance to cluster centres
    batch = batch.map(lambda x: calcDistCentre(x,centres.value))
    # Filter out instances that are near any cluster centre
    remain = batch.filter(all(x.dist>centres.radius))
    # Same operation as in Algorithm 1.1
    remain = remain.map(lambda x: calcDistModel(x,model.value))
    similar = remain.flatMap(lambda x: filterSimilarInstances(x,r))
    remove_instances = similar.collect()
    merged = similar.reduceByKey(add).map(lambda x: x/numInstances)
    updated = model.value - remove_instances + merged.collect()
    model = sc.broadcast(updated)
model' = model
```

Algorithm 2.1: Predict test instances (k-NN)

```
Input:
r = user-defined radius
k = user-defined k-number of neighbours
t = user-defined confidence threshold
test = test instances (in RDD)
model = broadcasted model

Output:
predictions = prediction for the test instances

Algorithm:
# Classify incoming instances based on distance to model instances
test = test.map(lambda x: calcDist(x,model.value))
prediction = test.map(lambda x: predict(x,r,k,t)).collect()
```

Algorithm 2.2: Predict test instances (k-Means)

```
Input:
r = user-defined radius
test = test instances (in RDD)
centres = broadcasted cluster centres
model = broadcasted model

Output:
predictions = prediction for the test instances

Algorithm:
# Classify incoming instances within radius of any cluster centre
test = test.map(lambda x: calcDistCentres(x,centres.value))
test = test.map(lambda x: predictWithinRadius(x,centres.radius))
# Classify incoming instances based on distance to model instances
test = test.map(lambda x: calcDistModel(x,model.value))
prediction = test.map(lambda x: predictByDistance(x,r,x.dist)).collect()
```

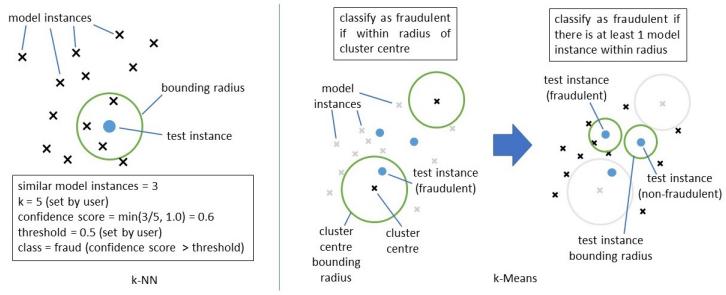

Fig. 3. An example of how test instances are classified for the k-NN and k-Means approaches.

A. Evaluation Approach

We identified two potential bottlenecks in our pipeline, namely the batch update in Phase 2 and the prediction operations. Hence, we will be evaluating the performance of each of these approaches.

Additionally, we calculate the accuracy for each approach using our validation dataset we reserved earlier, as well as the AUROC score with the test dataset. The reason AUROC is used instead of other metrics like AUPRC is because the ground truth of the test dataset is withheld from us, and the dataset authors utilised AUROC to measure the performance of an approach.

B. Dataset for Evaluation

As we have a limited number of data in our train and test datasets, we artificially generate additional transactions for them to increase the dataset size. To test the performance of our prediction operation, we simply increase the test dataset size by duplicating the test instances. This is because each test instance is applied the same prediction operation and we are only interested in the throughout of our prediction approach. Random sampling is also applied on the test dataset to generate smaller datasets for evaluation. In total, we generated 5 datasets of sizes 1,000, 5,000, 50,000, 500,000, and 5,000,000 respectively.

For our train dataset, we apply Synthetic Minority Over-Sampling Technique (SMOTE) [15] on the reserved instances to generate new unique fraudulent instances. In total, our reserved instances size is 100,000. We evaluate our batch update operation with the following batch sizes: 1, 50, 200, 1,000, 5,000, 10,000, and 100,000.

No modifications are done to the original datasets for evaluating the accuracy and AUROC score.

C. Results and Discussion

The results for our prediction approaches for both the k-NN and k-Means approaches are presented in Fig 4.

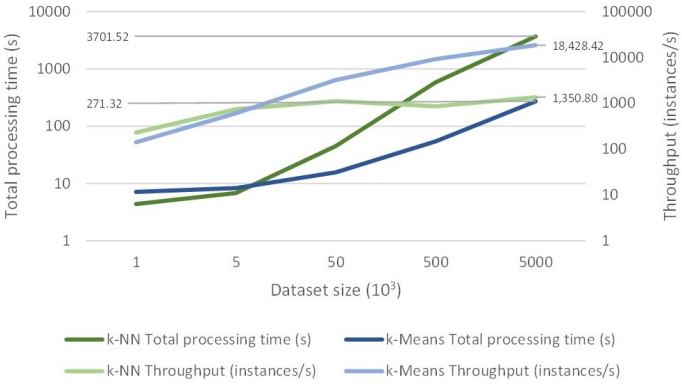

Fig. 4. The results for our prediction operation. The vertical axes are in logarithmic scale.

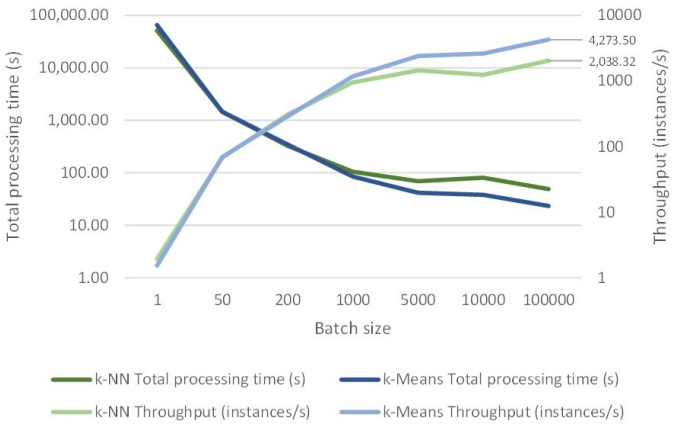

Fig. 5. The results for our batch update operation. The vertical axes are in logarithmic scale.

From our results, it appears that the processing time of the k-NN approach increases relatively linearly with the dataset size, while the k-Means approach appears to form a polynomial curve. The linear increase for the k-NN is expected as it is applying the same prediction operations for each test instance. And with the processing time scaling linearly with the dataset, the throughput of the k-NN approach stabilizes at approximately 1,350 instances for larger datasets. The polynomial increase in processing time observed for the k-Means approach, however, does not suggest that it is less efficient in terms of time complexity compared to the k-NN approach. Rather, this suggests that its processing time has yet to reach a consistent scaling. This is evidenced by the diminishing increase in its throughput, which still appears to be growing past the 5,000,000 dataset. Once the throughput stabilizes, it is expected to scale linearly. It appears that even with the polynomial increase in processing time, the k-Means approach outperforms the k-NN approach in terms of processing time and throughput. This is expected as less operations are done in the k-Means approach. Furthermore, as the throughput of the k-Means approach is expected to grow further, this suggests that it can handle larger datasets more efficiently compared to small datasets.

The results for our batch update approach in Phase 2 are presented in Fig 5.

It is observed that a smaller batch size results in a longer processing time for updating the model of an approach. This is because the number of read/write operations performed to retrieve the batches from the driver is high, which potentially bottlenecked the processing of new instances. As the batch size increases, there is a diminishing improvement in the processing speed as the decrease in number of read/write operations diminishes and the actual processing time of the inputs becomes the dominant factor. Alongside this, the throughput of both approaches also observed diminishing improvements as the batch size scales up. This is because the total time taken to process the entire test dataset defines how efficient an approach is. From the results, it appears that both approaches still see potential improvement in performance, although there are signs of it starting to plateau.

Based on our results, the k-NN approach is able to process up to approximately 1,350 instances per second for classifying unknown instances, while the k-Means approach can classify up to approximately 18,400 instances per second. This is more than 13 times the speed of the k-NN approach, which makes sense as a sizeable portion of the new instances are classified based on their distance to the cluster centres, leaving less unknown instances to be classified by the model instances, which there are less of compared to the k-NN approach. Furthermore, our results suggest that the k-Means approach can improve with a larger dataset as its throughput has yet to plateau.

Additionally, when processing batches of new fraudulent instances, the k-Means approach achieves more than twice the throughput of the k-NN approach at approximately 4,200 instances per second compared to approximately 2,000 instances per second. These results combined suggests that the k-Means approach is better suited to scale to big data in terms of speed.

The effectiveness of each approach in terms of accuracy (on the validation dataset) and AUROC score (on the test dataset) are presented in Table I.

TABLE I
RESULTS FOR EVALUATING EFFECTIVENESS OF BOTH APPROACHES.

	Accuracy (%)	**AUROC**	**Model size**
k-NN	34.69	0.4999	5912
k-Means	41.33	0.4993	4989[1]

[1]This is on top of k cluster centres.

As similarity-based approaches classify unknown instances based on their distance to known instances, the distribution between fraudulent instances and non-fraudulent instances must have a distinct separation. However, as shown by our empirical results, both approaches failed to properly classify the unknown instances. This suggests that the fraudulent and non-fraudulent data we are working with has a very high overlap, resulting in many false positives and false negatives.

Interestingly, both methods achieved an accuracy of lower than 50%, which is the accuracy rate of random guess. This

suggests that both approaches managed to learn and differentiate between fraudulent and non-fraudulent instances, but in the opposite manner to as was expected. This behaviour is known an anti-learning [16]. It appears that the k-NN approach achieved a higher learning rate as its accuracy diverges from the random guessing rate more than the k-Means approach. This suggests that the k-Means approach has more "misclassifications" compared to the k-NN approach. This makes sense as our k-Means approach classifies every new instance within radius of any of its cluster centres as fraudulent, which is not an effective classification method due to the overlapping distribution of fraudulent and non-fraudulent instances.

Despite the fact that both approaches managed to differentiate fraudulent and non-fraudulent instances to some degree, the AUROC scores suggest that both approaches are no better than random guessing. This is because it aggregates the trade-off between the true positive rate and the false positive rate of an approach for each decision threshold (this would be like the threshold confidence in our case). This means that this metric is very sensitive to imbalanced datasets, and is typically discouraged from being used in cases of imbalanced datasets. However, as we do not know the actual distribution of the data for the test dataset, no concrete conclusions can be drawn from this.

D. Summary of Results

Although the k-Means approach scales relatively well to large datasets, it compromises on its accuracy in classifying incoming instances. Based on our empirical results, the k-Means approach sees a decrease of approximately 7% in accuracy compared to the k-NN approach, while achieving more than 10 times the improvement in processing speed. This would usually be not too bad of a trade-off, especially when processing speed is prioritized. However, as both approaches are not very accurate to begin with, this may be a rather costly compromise.

Nevertheless, based on our results, it appears that some similarity-based approaches have potential scalability in the big data sense, especially if the model size is small. This suggests that the model size is the limiting factor as more computation is done when classifying unknown instances.

V. Conclusion

In this project, we evaluated two simple similarity-based classification approach in the big data sense, namely the k-NN approach and the k-Means approach. We adopted both approaches and modified them to suit one-class classification tasks which allows us to significantly reduce the number of train instances to handle. Based on our empirical results, it appears that there are merits and disadvantages for both approaches. The k-Means approach achieves very promising scalability to large amounts of data while compromising on its accuracy in classifying unknown transactions. The opposite is observed for the k-NN approach.

References

[1] K. Sk and R. Vadlamani, "Credit card fraud detection using big data analytics: Use of psoaann based one-class classification," pp. 1–8, 08 2016.

[2] N. MUSTAFA, J.-P. LI, R. A. Memon, and M. Z. Omer, "A classification model for imbalanced medical data based on pca and farther distance based synthetic minority oversampling technique," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 1, 2017.

[3] Q. Wei, Y. Ren, R. Hou, B. Shi, J. Y. Lo, and L. Carin, "Anomaly detection for medical images based on a one-class classification," in *Medical Imaging 2018: Computer-Aided Diagnosis* (N. Petrick and K. Mori, eds.), vol. 10575, pp. 375 – 380, International Society for Optics and Photonics, SPIE, 2018.

[4] J. Rodríguez-Ruiz, J. I. Mata-Sánchez, R. Monroy, O. Loyola-González, and A. López-Cuevas, "A one-class classification approach for bot detection on twitter," *Computers & Security*, vol. 91, p. 101715, 2020.

[5] B. Krawczyk, M. Woźniak, and F. Herrera, "Weighted one-class classification for different types of minority class examples in imbalanced data," in *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pp. 337–344, 2014.

[6] S. Zhang, Z. Deng, D. Cheng, M. Zong, and X. Zhu, "Efficient knn classification algorithm for big data," *Neurocomputing*, vol. 195, 02 2016.

[7] X. Chen and D. Cai, "Large scale spectral clustering with landmark-based representation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, Aug. 2011.

[8] G. Chen, Y. Liu, and Z. Ge, "K-means bayes algorithm for imbalanced fault classification and big data application," *Journal of Process Control*, vol. 81, pp. 54–64, 2019.

[9] Y. Sahin and E. Duman, "Detecting credit card fraud by decision trees and support vector machines," *IMECS 2011 - International MultiConference of Engineers and Computer Scientists 2011*, vol. 1, pp. 442–447, 03 2011.

[10] K. Fu, D. Cheng, Y. Tu, and L. Zhang, "Credit card fraud detection using convolutional neural networks," in *ICONIP*, 2016.

[11] D. Tax and R. Duin, "Uniform object generation for optimizing one-class classifiers," *Journal of Machine Learning Research*, vol. 2, 04 2002.

[12] P. Zheng, S. Yuan, X. Wu, J. Li, and A. Lu, "One-class adversarial nets for fraud detection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1286–1293, Jul. 2019.

[13] X. Min and R. Lin, "K-means algorithm: Fraud detection based on signaling data," in *2018 IEEE World Congress on Services (SERVICES)*, pp. 21–22, 2018.

[14] Y. Dai, J. Yan, X. Tang, H. Zhao, and M. Guo, "Online credit card fraud detection: A hybrid framework with big data technologies," in *2016 IEEE Trustcom/BigDataSE/ISPA*, pp. 1644–1651, 2016.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, p. 321–357, Jun 2002.

[16] A. Kowalczyk and O. Chapelle, "An analysis of the anti-learning phenomenon for the class symmetric polyhedron," in *ALT*, 2005.