A Cost-Sensitive Cosine Similarity K-Nearest Neighbor for Credit Card Fraud Detection

Sara Makki*[§], Rafiqul Haque[†], Yehia Taher[‡], Zainab Assaghir[§], Mohand-Saïd Hacid* and Hassan Zeineddine[§]

*Laboratoire LIRIS, Université de Lyon, Villeurbanne, France
[†]Cognitus Centre for Big Data Science R&D, Paris, France
[‡] Laboratoire DAVID, Université de Versailles, Versailles, France
[§]Lebanese University, Beirut, Lebanon

***Abstract*—Credit card fraud commonly happens in financial institutes such as banks. Fraud results in a huge financial damage that may reach to billions of dollars every year. Detecting and preventing credit card fraud manually is a labor intensive and relatively ineffective approach. Therefore, a significant effort was made to develop automated solutions for fraud detection. Researchers dedicated their works on designing and developing models and systems in particular, the fraud anlaysis systems that enable to detect different types of fraud in different sectors including insurance, telecommunication, financial audit, financial markets, money laundering, credit card, *etc.* However, some problems remains unsolved. Of all, the most prevalent one is the extreme class imbalance. In this paper, we aimed at addressing this problem. We focused on the K-Nearest Neighbor (KNN) classifier and investigated the cost-sensitive approaches used for KNN. Also, we presented a novel cost-sensitive KNN approach that we developed using Cosine Similarity (CoS). We compared our model with the other methods to verify its efficiency, and we proved using several performance measures that it's a better approach than other KNN algorithms.**

***Index Terms*—Fraud detection, K-Nearest Neighbor, Imbalanced classification, Cosine similarity, Cost-sensitive learning.**

I. Introduction

Credit card fraud has always been a huge interest to financial institutes and credit card users due to its detrimental effect in case of fraudulent event. Credit card fraud is defined as an unauthorized use of a credit card account. Fraudster uses credit card information, often remotely, without the knowledge of the card owner or the card issuer. There are different types of frauds. *Simple Theft (offline fraud)* occurs if the card is stolen; *application fraud* occurs when credit card applicants obtain new credit cards using fake identity or information; *bankruptcy fraud* sometimes is a result of application fraud; finally, *counterfeit fraud* occurs when only the details of a legitimate card are stolen (skimming or shoulder surfing) and used remotely (mobile sales, online, *etc.*). Detecting and preventing these frauds are challenging for financial institutes. Human-driven fraud detection approaches are labor intensive.

Over the years, researchers have been working on developing an automated fraud detection system to reduce and prevent frauds specifically credit card fraud (*e.g.,* [1], [2], [3]). However, the major challenge is the *skewed data distribution* also known as the class imbalance. In this problem, the dataset is extremely imbalanced and highly skewed [4]. This problem can be summarized as having legitimate transactions examples in the training set a lot more than fraudulent ones. This phenomenon makes a balanced training very difficult and detection of fraud will be more challenging due to inadequate training examples. In the case of credit card fraud, the imbalance is often extreme (the fraudulent transactions in the training set are less than 10%). Several machine learning methods were investigated for the credit card fraud detection and for class imbalance problem. A significant research works have been done concerning this problem (*e.g.,* [5], [6], [7]). However, an efficient solution is still missing in state of the art. The available imbalance classification approaches that are used, often increase the detection of the minority group at the cost of generating false predictions for the other class, which leads to an overall decrease of the model's accuracy.

Our Contribution: In this paper, we focused on the K-Nearest Neighbor (KNN) classifier. We investigated the cost-sensitive approaches used for KNN and presented a new cost-sensitive KNN approach that we developed using Cosine Similarity (CoS). We compared our model with the other methods to verify its efficiency, and we proved using four performance measures (accuracy, sensitivity, PR curve and the F_1 score) that it's a better approach than other KNN algorithms.

The remainder of the paper is organized as follows. In Section II, we review the related work to credit card fraud using KNN and imbalanced classification specially the cost-sensitive approaches. In Section III, we describe briefly the KNN classifier and the methods compared, then in Section IV we will describe the approach we developed. In Section V, we describe the data provided and we present the results and finally a conclusion and future work.

II. Related Work

Many studies investigated the skewed data distribution in classification, and introduced ways to tackle this problem. There are three different ways to address the class imbalance

This work partially benefited from the support of the Lebanese National Council for Scientific Research (CNRS-L).

issue [8]. The first one is on data-level, as a preprocessing step to balance classes. Usually over- or under-sampling are done before applying a machine learning algorithm. The second one is an algorithm-level approach, like cost-sensitive learning or one-class classification, that alter the original algorithm to focus on the minority group. The basic idea behind cost-sensitive models is to give higher weights to the small class. It's equivalent to assigning higher costs to false negatives. However, in the one-class classification, the training is done using only one class, usually the minority group. The third approach consists of a combination of the two previous approaches.

As for credit card fraud detection, researchers used different machine learning algorithms to classify credit card transactions as normal or fraudulent. A survey of these methods and their application to credit card datasets is presented by Tripathi and Pavaskar [9] and SamanehSorournejad [10].

KNN was rarely investigated for credit card fraud detection. Ganji and Mannem [11] proposed a credit card fraud detection system using a data stream outlier detection algorithm which is based on Reverse k-nearest neighbors. Whitrow et al. [12] considered transaction aggregation and showed that it is effective, and compared multiple methods including K-nearest neighbors, support vector machines and logistic regression, and they concluded that random forest outperforms all of them.

Credit card fraud detection is a common example of imbalanced data classification problem. Sahin et al. [13] and Bahnsen et al. [14] proposed a cost-sensitive decision tree approach that splits the data by minimizing the sum of misclassification costs. The authors compared their approach to well-known methods using a real credit card fraud dataset.

Kamaruddin and Vadlamani [15] proposed using one-class classification approach to solve the imbalance problem. They proposed a hybrid system of Particle Swarm Optimization and Auto-Associative Neural Network (PSOAANN), and implemented it in a Spark computational framework.

Qibei et al. [16] proposed Imbalance Class Weighted Support Vector Machine (ICW-SVM) to detect credit card fraud, according to the imbalance characteristics of the data, after reducing its dimension using Principal Component Analysis (PCA). They applied their model and proved its efficiency on a real bank dataset.

III. Methods

In this section, we will describe the KNN classifier algorithm using simple voting or distance weighted KNN and a cost-sensitive KNN approach that was introduced by Qin et al. [17]. We will also compare these methods with a cost sensitive decision tree algorithm and a one class classification support vector machine.

A. K-Nearest Neighbour (KNN)

KNN is a data mining method widely used for classification and regression. It is a simple algorithm that consists of using the k nearest points to the one we aim to predict [18]. A specific norm is used to measure the distance between points. The norm commonly used to measure the distance between two observations p and $q \in \mathbb{R}^n$ is the euclidean distance:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

The KNN algorithm also depends on a distance rule for the classification using the k nearest neighbors of a new sample.

1) Simple Voting: The most straightforward rule used for the classification is the simple voting. The new observation is assigned to the class of the majority of the k nearest points. The classification is done according to the following formula:

$$\hat{y} = \underset{c \in Y}{\operatorname{argmax}} \sum_{k \in K} \delta(c, class(k))$$

Where $\hat{y}$ represents the foretasted value of a new observation y, c represents the possible classification categories, and K is the subset of the chosen nearest neighbors of y. The function δ is defined as follows:

$$\delta(x,y) = \begin{cases} 0 & \text{if} \quad x = y \\ 1 & \text{if} \quad x \neq y \end{cases}$$

The function argmax returns the value of the category 0 or 1 at which the maximum is reached, i.e. the category of the majority of the neighbors.

2) Distance Weighted: Another method is a distance weighted voting. The idea behind this approach is to take into account the distance of the neighbors and to assign a higher weight to the closest ones. The prediction is done as follows:

$$\hat{y} = \underset{c \in Y}{\operatorname{argmax}} \sum_{k \in K} w_k \delta(c, class(k))$$

$$w_k = \frac{1}{d_k^2}$$

Where w_k is the assigned weight and d_k is the euclidean distance between y and the considered neighbor.

B. Cost Sensitive KNN

In [17], the authors introduced two cost-sensitive approaches for KNN. The first one called Direct Cost-Sensitive KNN (DirectCS-KNN) is based on calculating class probabilities from the original KNN algorithm using the following formula:

$$P_i = \frac{k_i}{k}$$

Where P_i is the probability that the class of a new observation y is i, and k_i is the number of neighbors of class i.

The other approach was distance weighted called Distance-CS-KNN. Considering two categories 1 and 0, the idea consists of calculating costs for both classes C_1 and C_0 as follows:

$$C_1 = fp \times P_0 \quad \text{and} \quad C_0 = fn \times P_1$$

fp and fn represents respectively the costs of false positives and false negatives. The new sample is assigned to the class with lower C_i. In this case, the P_i are calculated in a cost-sensitive manner different than the one in DirectCS-KNN.

$$P_1 = \frac{w_1}{w_1 + w_0} \quad \text{and} \quad P_0 = \frac{w_0}{w_1 + w_0}$$

Where w_i are the cost sensitive weights calculated as shown here:

$$w_i = \sum_{j=1}^{ki} w_j$$

C. Cost-Sensitive Decision Tree (C5.0)

C5.0 is one of the most common decision tree algorithms using cross entropy and information gain to create the partition and the splits of the tree. The costs are implemented to the decision boundaries, not in the training algorithm [19], so the new decision boundary considered for classifying an observation into class 1 is:

$$\frac{p_1}{p_0} > \frac{C_{1/0}}{C_{0/1}} \; \frac{\pi_0}{\pi_1}$$

Where π_i is the prior probability of an observation to be in class i. p_i and $C_{j/i}$ are respectively the estimator of the probability and the cost of wrongly classifying an observation of class i as j.

D. One Class Classification Support Vector Machine

Using only the minority cases, a one-class classification SVM is applied with the aim of learning only the characteristics of this class [20].
The purpose of this method is to find a "small" region that groups most of the minority training observations. This requires defining a function f that returns 1 if a point belongs in this region and -1 elsewhere [21]. This function is found by optimizing the following problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}||w||^2 + \frac{1}{\nu l}\sum_{i=1}^{l} \zeta_i - \rho \\ \text{subject to} \quad & \left\{ \begin{array}{l} w^T \Phi(x_i) \geq \rho - \zeta_i \\ \zeta_i \geq 0, i = 1, \ldots, l \end{array} \right. \end{aligned}$$

Where $\Phi : X \to F$ is a mapping function. This problem is solved by separating the points from the origin with maximum margin ($\frac{\rho}{||w||}$) using a hyperplane of equation: $w^T \Phi(x_i) = \rho$.

IV. Cost-sensitive KNN approach (CoSKNN)

We developed a Cost-Sensitive KNN approach. Our aim was to tackle the imbalance problem by using cosine similarity as a distance metric and by introducing a score for the classification. In order to improve the method's performance in terms of imbalance, we also studied the choice of the score's thresholds and the number of neighbors to consider. These steps are described in the following:

Step 1: The use of Cosine Similarity

The first step of our new approach is the change of the distance metric used. This metric consists of calculating the angle's cosine between two vectors p and q representing two observations:

$$CoS\ (p, q) = \frac{\sqrt{\sum_{i=1}^{n} p_i q_i}}{\sqrt{\sum_{i=1}^{n} p_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}}$$

We replaced euclidean distance used in KNN with this metric when calculating the distance between observations in order to find the nearest neighbors.

Note that, this metric ranges between -1 and 1. If CoS is close to 1, it indicates that the angle between the two observations is close to zero and therefore they are similar (neighbors).

The advantage of using CoS instead of euclidean distance is highlighted in the coming section when we compared KNN (whether simple voting or with weighted distance) with both metrics and we found that CoS is better in terms of sensitivity.

Step 2: Introducing the score S_y

The second step of our approach, after finding the neighbors, is to introduce an imbalanced classification approach while using CoS. The idea is to evaluate the similarity of an observation to its neighbors of the minority class, taking into account the other class as well. This was done by calculating the following score S_y for each sample y:

$$S_y = \frac{\sum_{i=1}^{k} C_i \ . \ CoS_i}{\sum_{i=1}^{k} CoS_i}$$

Where:
• k is the number of neighbors considered for y.
• C is a vector of length k, $C_i \in \{0, 1\}$ that represents the classes of the neighbors. 0 is used to denote the class of the majority group and 1 is used for the minority group.
• CoS is another vector representing the Cosine similarity between y and its neighbors.
This score ranges from 0 to 1. It works similarly to a probability, that describes the likelihood of an observation to be in the minority group. When it's close (or equal) to zero, it indicates that the neighbors are mostly (or all) of the majority group, which would lead to a majority group classification. However, when at least one of the neighbors of y is of the minority class, this score will be higher than zero; and closer to one, the more the neighbors are of the minority group.

A. The classification and the score thresholds

The classification is done according to a certain threshold $\alpha \in [0, 1]$.

$$\hat{y} = \left\{ \begin{array}{lll} 0 & \text{if} & S_y \leq \alpha \\ 1 & \text{if} & S_y > \alpha \end{array} \right.$$

The choice of α is not straightforward. A very low value will lead to a large number of false positives (observations of the majority group classed as minority). However, a high threshold value will lead to very low sensitivity rate. Therefore, α should have a slightly low value. Taking into account the imbalance ratio of $\approx 5\%$, α was later chosen according to the 95^{th} percentile of S_y, and optimized by comparing the sensitivity according to the values of α.

B. The choice of k

The choice of k has an effect on many aspects of the approach. The number of neighbors should be large enough to be informative about the sample's neighborhood.
On the other hand, due to the imbalance, a high number of neighbors will make the classification biased towards the majority group and time consuming.
After trying several possible values of k, we found that considering 10 neighbors is the best for our case.

V. Experiment and Results

To prove the efficiency of our approach, we compared it to the original KNN classifier using both euclidean distance and cosine similarity, with simple voting and distance weighted approach. We also compare it with cost sensitive C5.0, one class classification SVM and Distance-CS-KNN [17].

A. Data Description

The data we used is extracted from a credit card fraud labeled data[1]. The data contains 10 million observations (in our case, the observations are financial transactions) and 8 variables described in the following.
The explanatory variables are: *gender* representing the client's gender, *state* is a categorical variable denoting the state where the client lives, *cardholder* representing the number of cards that the client have, *balance* is a continuous variable that indicates the balance on the credit card, the number of transactions and international transactions made to date, respectively denoted by *numTrans* and *numIntTrans*, and *creditLine* which is the credit limit of the client, and the variable to predict *fraudRisk* indicating if each observation is fraud (denoted 1) and legitimate (denoted 0). 596,014 (representing 5.96%) are fraud cases and 9,403,986 (representing 94.03%) are legitimate.

In this paper, we extracted a part of the original data due to the time consumption of the KNN method when calculating the distance to all observations in the training set. The new data consists of 6000 credit card transactions (observations) in which we have 5657 *Legitimate*(0) cases, and 343 *fraud*(1). This dataset takes into account the imbalance ratio and have the same characteristics as the original one.
These proportions exemplify the extreme imbalance. The fraud cases represents 5.7% of the dataset. The fraud detection in this case is very challenging. In fact an accuracy rate less than 94.3% is not acceptable, because simply an algorithm that classify all data points as legitimate will give us this high accuracy. The choice of other performance measures will be discussed later. This data is divided between train (3999 transactions) and test (2001 transactions) with similar ratio of imbalance. For all KNN methods, 10 nearest neighbors are considered to classify the new samples, and data is first normalized using the mean and standard deviation of the variables, to avoid bias towards variables with large ranges [18], because the explanatory variables are on widely different scales.

[1]Available at http://packages.revolutionanalytics.com/datasets/

B. Results and Discussion

In this section, we show the results of the comparison of our approach with the other existing in the state of the art. The compared methods are given in Table I.

TABLE I
COMPARED METHODS AND THEIR DESCRIPTION

Abbreviation	Method's Description
EucKNN	Simple voting KNN using the euclidean distance
CKNN	Simple voting KNN using cosine similarity
DEucKNN	Distance weighted KNN using euclidean distance
DCKNN	Distance weighted KNN using cosine similarity
Distance-CS-KNN	The distance based cost sensitive approach introduced by Qin et al. [17]
CS - C5.0	Cost sensitive C5.0 decision tree
OCC - SVM	One Class classification Support Vector Machine
CoSKNN	Cost sensitive cosine similarity based KNN

The use of the accuracy alone as a performance measure is misleading. Other measures should be considered as well, like the sensitivity, also known as recall. The challenge is that most imbalanced classification methods focus on increasing the sensitivity, which will lead to a slight decrease in accuracy. This decrease that can sometimes be just 1% may seem insignificant, but in fact, it hides a high number of false alarms. Thus, we need to rely also on measures that finds a trade-off between the high accuracy and the sensitivity; the Area Under Precision-Recall Curve (AUPRC), is used for that purpose. However, since we may not always be able to plot this curve like the case of CS - C5.0 and OCC - SVM, we will also use the F_1 score, also known as F-measure. To evaluate these measures, a confusion matrix (Table II) is calculated using the test set.

TABLE II
FORM OF CONFUSION MATRIX

Predicted	Actual	
	Legitimate (0)	Fraud (1)
Legitimate (0)	True Negative (TN)	False Negative (FN)
Fraud (1)	False Positive (FP)	True Positive (TP)

$$Accuracy = \frac{TP + TN}{TN + FN + TP + FP} \times 100$$

The accuracy shows the percentage of the legitimate and fraud transactions correctly predicted.

$$Recall = \frac{TP}{TP + FN} \quad \text{and} \quad Precision = \frac{TP}{TP + FP}$$

However, the recall or sensitivity focus only on the fraud detection and the true positive rate, and the precision measures the fraction of examples classified as positive that are truly positives.

The Precision-Recall (PR) curve is obtained by plotting the precision over recall rate through different class probabilities thresholds. The closer the curve is to the upper-right-hand corner the better the model is. It is not always straightforward to find the class probabilities, so we will also use the F_1 score. The higher this score the better.

$$F_1 \text{ score } = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Table III shows the performance measures (the accuracy, the sensitivity, the AUPRC and the F_1 score) for all methods and our new approach. The PR curves are shown in the Figure 1.

TABLE III
TABLE SUMMARIZING THE PERFORMANCE MEASURES

Method	Accuracy	Sensitivity	AUPRC	F_1 score
EucKNN	95.8%	0.30	0.39	0.45
CKNN	95.8%	0.42	0.53	0.53
DEucKNN	95.6%	0.32	0.22	0.46
DCKNN	95.4%	0.34	0.54	0.46
Distance-CS-KNN	94.5%	0.53	-	0.52
CS - C5.0	93.3%	0.65	-	0.53
OCC - SVM	88.9%	0.22	-	0.18
CoSKNN	95.5%	0.51	0.54	0.56

This table shows that the accuracy is higher than 94.3% for all models except the CS - C5.0 and OCC - SVM. The slight differences of the accuracy between all the other methods shows how much information this measure hides when the imbalance is extreme. We can conclude from the table when comparing the performance measures of EuCKNN with CKNN and DEuCKNN with DCKNN that the use of cosine similarity is improving the classification according to the sensitivity, AUPRC and F_1 score, with a reasonable decrease in accuracy.

Our approach CoSKNN is outperforming all the methods according to the AUPRC and F_1 score. It is considerably improving the sensitivity when compared to the simple KNN. The other cost-sensitive models are performing better in terms of sensitivity, but at the cost of raising false alarms and decreasing the accuracy sometimes to a less than acceptable value, like the case of CS-C5.0 and OCC-SVM.

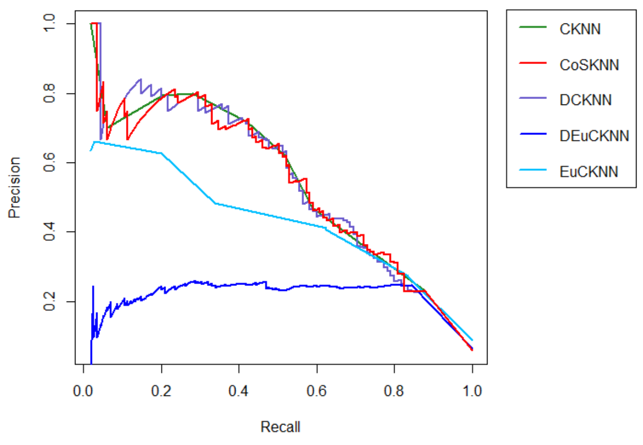

Fig. 1. PR curves for all methods

VI. CONCLUSION AND FUTUR WORK

Credit card fraud detection is one of the most challenging problems for financial institutes. The financial institutes such as banks are losing billions of dollars every year due to fraudulent activities committed by the fraudsters. Over the years, a number of solutions has been been proposed in large bodies of literature. However, some problems are still open. The class imbalance problem is one of the most critical problems that is yet to be solved.

In this paper, we aimed at addressing this problem. We investigated the use of KNN in fraud detection. We proposed a cost-sensitive KNN approach to tackle the imbalance problem. We provided a comprehensive detail of our new approach. In our approach we used cosine similarity instead of the euclidean distance to find the neighbors, and then calculated a certain score to evaluate the probability of fraud risk.

We also presented a comparative study in this paper. In our study, we compared the performance of simple voting KNN and distance weighted KNN using both euclidean distance and cosine similarity, with another cost sensitive KNN, decision tree approach, one class classification SVM and our new approach. The comparison was done by applying these methods to a credit card fraud dataset with imbalance, using multiple performance measures, mostly relying on AUPRC and F_1 score. This experiment shows that our approach is outperforming all the other methods.

We encountered several challenges in our study. The most prevalent one is the *elapsed time* that restricted us the use of number of observations for the training set which could not exceed 3999. This is an obvious limitation of our experiment.

Several works have been lined up to extend our current work. We planned to work on implementing our approach in a big data environment in order to use the massive amount of data. Another interesting task which we planned is finding an optimized threshold α that can be selected automatically instead of letting user to investigate and find it.

References

[1] P. Richhariya and P. K. Singh, “Evaluating and emerging payment card fraud challenges and resolution,” *International Journal of Computer Applications*, vol. 107, no. 14, 2014.

[2] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, “Data mining for credit card fraud: A comparative study,” *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, 2011.

[3] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi, “Learned lessons in credit card fraud detection from a practitioner perspective,” *Expert systems with applications*, vol. 41, no. 10, pp. 4915–4928, 2014.

[4] C. Phua, D. Alahakoon, and V. Lee, “Minority report in fraud detection: classification of skewed data,” *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 50–59, 2004.

[5] Z.-H. Zhou and X.-Y. Liu, “Training cost-sensitive neural networks with methods addressing the class imbalance problem,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2006.

[6] S. Ertekin, J. Huang, and C. L. Giles, “Active learning for class imbalance problem,” in *Proceedings of the 30th annual international ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2007, pp. 823–824.

[7] M. Wasikowski and X.-w. Chen, “Combating the small sample class imbalance problem using feature selection,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1388–1400, 2010.

[8] B. Krawczyk, “Learning from imbalanced data: open challenges and future directions,” *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[9] K. K. Tripathi and M. A. Pavaskar, “Survey on credit card fraud detection methods,” *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 11, pp. 721–726, 2012.

[10] S. Sorournejad, Z. Zojaji, R. E. Atani, and A. H. Monadjemi, “A survey of credit card fraud detection techniques: Data and technique oriented perspective,” *CoRR*, 2016.

[11] V. R. Ganji and S. N. P. Mannem, “Credit card fraud detection using anti-k nearest neighbor algorithm,” *International Journal on Computer Science and Engineering (IJCSE)*, vol. 4, no. 6, pp. 1035–1039, 2012.

[12] C. Whitrow, D. Hand, J. Juszczak, D. Weston, and N. M. Adams, “Transaction aggregation as a strategy for credit card fraud detection,” *Data Mining and Knowledge Discovery*, pp. 30–55, 2009.

[13] Y. Sahin, S. Bulkan, and E. Duman, “ A cost-sensitive decision tree approach for fraud detection,” *Expert Systems with Applications*, vol. 40, pp. 5916–5918, 2013.

[14] A. C. Bahnsen, A. Stojanovic, D. Aouada, and B. Ottersten, “Cost sensitive credit card fraud detection using bayes minimum risk,” in *Proceedings of the 2013 12th International Conference on Machine Learning and Applications*, vol. 1, 2013, pp. 333–338.

[15] S. Kamaruddin and V. Ravi, “Credit Card Fraud Detection using Big Data Analytics : Use of PSOAANN based One-Class Classification,” *Proceedings of the International Conference on Informatics and Analytics*, pp. 33:1–33:8, 2016.

[16] Q. Lu and C. Ju, “Research on credit card fraud detection model based on class weighted support vector machine,” *Journal of Convergence Information Technology*, vol. 6, pp. 62–68, 2011.

[17] Z. Qin, A. T. Wang, C. Zhang, and S. Zhang, “Cost-sensitive classification with k-nearest neighbors,” in *Knowledge Science, Engineering and Management*. Springer Berlin Heidelberg, 2013, pp. 112–131.

[18] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. Morgan Kaufman Publishers, 2003.

[19] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. Springer, 2013.

[20] A. Nicholas and R. Daniel, “One-class support vector machines: Methods and applications,” 2008.

[21] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the support of a high-dimensional distribution,” *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001.