Chapter 4
Modifying the SMOTE and Safe-Level SMOTE Oversampling Method to Improve Performance

Kgaugelo Moses Dolo and Ernest Mnkandla

4.1 Introduction

The binary class distribution is a classification problem that involves two classes of the dataset. For a machine learning data model to generate accurate model outcome, the two classes of the dataset must be equally represented to avoid the issue of biasness [1–3]. Thus, oversampling and under-sampling are two techniques required to balance the class distribution of datasets in cases where the class distribution is skewed [1–4]. In this study, the original shape of the dataset represents the true context of the problem, which means that oversampling of the minority class and down-sampling of the majority class must be done to preserve the original shape of the dataset.

In general, the random selection of data observations from a population that has duplicates can result in a sample dataset that has duplicates. The implication is that, random sampling of a majority class that has duplicate data observations will result in duplicate data observations. It therefore means that, random sampling of majority class alone may not solve the problem of skewed class distribution for a binary classification problem. It is therefore suggested in this chapter that when down sampling the majority class, a random sampling on the majority class must be performed on a dataset that contains no duplicate data instances.

K. M. Dolo (✉) · E. Mnkandla
University of South Africa, Pretoria, South Africa
e-mail: 58527141@mylife.unisa.ac.za; mnkane@unisa.ac.za

I. Woungang, S. K. Dhurandher, *4th International Conference on Wireless, Intelligent and Distributed Environment for Communication*, Lecture Notes on Data Engineering and Communications Technologies 94, https://doi.org/10.1007/978-3-030-89776-5_4

For a highly skewed data distribution, re-oversampling of the minority class that negatively affects the original shape of the minority class data distribution must be avoided. Re-oversampling must be done to introduce new possibilities of the minority class rather than duplicating the existing minority class data instances. SMOTE is an oversampling method that is known to generate new instances of the minority class rather than duplicating the existing data instances of the minority class [5].

SMOTE oversamples the minority class by computing median feature vectors between a nominal feature sample and its potential nearest neighbours by Euclidean distance of standard deviations [5]. Duplicating minority training samples to reduce biasness of data distribution introduces high variance of data distribution. Thus, SMOTE is an important oversampling method that reduces variance of data distribution. The synthetic instances generated influence a classifier model to create less specific decision regions which are smaller.

Furthermore, positive influence can be better learned in general regions than positive instances subsumed around negative instances, which means that the SMOTE method suffers from a problem of generalization, whereby the region of a majority class is blindly generalized without considering the majority class [6, 7]. The generalization problem of the SMOTE method is particularly visible in a case of highly skewed class distribution since the minority class is thinly scattered in relation to the majority class. Thus, the probability of class mixture is very high. To keep the SMOTE method efficient and effective, an improvement of the algorithm is required.

Borderline SMOTE is an oversampling method designed to improve the performance of SMOTE oversampling method [1, 6]. The performance is improved by separating the positive instances into three regions namely borderline $\left(\frac{1}{2}K \leq n < K\right)$, noise $(n = k)$ and safe $\left(0 \leq n < \frac{1}{2}K\right)$ [1, 6]. The three regions are separated while considering the negative instances on the K Nearest Neighbours. The borderline SMOTE uses the same oversampling method as SMOTE.

However, borderline SMOTE oversamples only the borderline instances of the minority class rather than oversampling entire instances of the minority class. Logically, the two consecutive instances are obviously not different, but they are divided into two regions (noise and borderline), whereby the first instance is selected for oversampling and the other instance is declined for oversampling.

Another method is Safe-Level SMOTE, which is an oversampling method that creates safe-level synthesis of the minority class [1, 6, 7]. The synthetic instances are placed closer to the safe level; that is to say, the safe level closer to K is nearly noise. The safe level is defined as the number of positive instances within K Nearest Neighbour but not equal to K Nearest Neighbour [6]. The Safe-Level SMOTE is therefore found to be a promising algorithm with a positive impact in this study.

The rest of the chapter is organized as follows. Section 4.2 presents the research questions addressed by this work and their objectives. Section 4.3 presents the Modified SMOTE method. Section 4.4 presents the simulation results, and Sect. 4.5 concludes the chapter.

4.2 Research Questions and Objectives

4.2.1 Research Questions

Since the banking sector is migrating from data analytics to data insights in South Africa [8], the study therefore sets out to explore the following main research question: Will the oversampling of the minority class that generates new data instances at the safe level of the minority class and under-sampling of the majority class that randomly selects non-duplicate data instances preserve the nature of the dataset and the original context of the problem for credit card fraud data when dealing with highly skewed class distributions? This main research question is broken down into the following sub-questions:

(a) Does Safe-Level SMOTE oversampling method (on minority classes) used with under-sampling method (that eliminates duplicate data samples on majority class) have positive impact on reducing the high skewedness of the class distribution than SMOTE oversampling method (on minority classes) used with under-sampling method (that also eliminates duplicate data samples on majority class)?
(b) Do Safe-Level-SMOTE oversampling method and the under-sampling method (that eliminates duplicate data samples on majority class) reduce or eliminate the problem of overlapping data samples between fraudulent and non-fraudulent classes?

This study therefore sought to answer these questions by carrying out the objectives presented in the next section.

4.2.2 Research Objectives

The main objective of this study is to obtain a balanced class distribution of credit card fraud data that preserves the nature of the dataset and the original context of the problem. The main objective is further broken down into two sub-objectives presented below:

(a) To determine if modified Safe-Level SMOTE oversampling method used with under-sampling method has a positive impact on reducing the high skewedness of the class distribution than the modified SMOTE oversampling method used with under-sampling method. This objective will be implemented by developing and running a modified SMOTE algorithm and a modified Safe-Level SMOTE algorithm on the minority class data, and an under-sampling algorithm on the majority class data.
(b) To investigate if the modified Safe-Level SMOTE oversampling method and the under-sampling method reduce or eliminate the problem of overlapping data samples between fraudulent and non-fraudulent classes. This objective will be

implemented by running a modified Safe-Level SMOTE on minority class data and an under-sampling algorithm on majority class data.

The next section discusses the research design followed to achieve the research objectives of this study.

4.3 Research Design

The modified SMOTE method in this study will be compared to the SMOTE method, and the modified Safe-Level SMOTE method will be compared to the Safe-Level SMOTE. The modified SMOTE and the modified Safe-Level SMOTE were used to oversample the data observations of the minority class of the dataset together with down-sampling method that removes duplicates and randomly chooses non-duplicate data samples from the majority class in order to control the best fit line to an optimum place that represents fraudulent transactions and non-fraudulent transactions equally. A dataset with equal representation of fraudulent transactions and non-fraudulent transactions makes it easier for a classification model to learn data of the two classes well. Thus, the modified SMOTE and modified Safe-Level SMOTE were tested by running Artificial Neural Network, Support Vector Machine, Naïve Bayesian and k-Nearest Neighbours algorithms.

While the objective of the SMOTE algorithm is to generate a new data instance between two existing data instances of the minority class [5]. This chapter argues that the logic used to generate new data instances by the SMOTE does not always generate a new data instance between two given data instances.

From the **for loop** of attributes to the above SMOTE Algorithm 1, the algorithm is looping through the attributes of the dataset. The synthetic instance is generated by finding the difference of attribute values between the data instance of interest and its chosen nearest neighbour. The difference is then multiplied by the gap which is the random value chosen between 0 and 1. The multiplication of the difference and the gap is added to the data instance of interest.

Although this method works well, there are limitations to its objective. The limitations are introduced by the logic behind the mathematics of the discussed attribute loop of the SMOTE method. This chapter therefore suggests that if two data instances have values of opposite signs, then the SMOTE algorithm will not generate a new data instance that is between the two data instances, which violates the objective of SMOTE algorithm. The difference in attribute loop of SMOTE will change to addition, given that the two data instances have values of opposite signs.

Below is a demonstration of the above claim using the attribute loop of SMOTE. The attribute loop of SMOTE is where the new data instance is generated between two data points. Figure 4.1 shows the function of attribute loop of SMOTE.

The following random values of instances will be chosen to test the claim made in this study: instances = [[−10, −21, −4, −45, −66, −93, 1, 10, 21, 4, 45, 66, 93, 1]]. To pictorially see the output, the x-axis values are required. The standard

```
Algorithm 1: SMOTE
Input: Minority data observations-T, Size of new
instance in percentage-N,
     Size of the nearest neighbours-K
  Output: I=(N/100)*count(T) - is the newly generated
minority class instances of
                size I - Synthetic-instances.
  Begin
   1. No-of-minority = count(T)
   2. No-new-Instances = I
   3. No-of-attributes = count(T[0] - 1
   4. Minority-instances = T
   5. Synthetic-instances= []
   6.
   7. For positive-instance in positive-instances:
   8.       Compute 5 neighbours of positive-
      instance and call it nnarray.
   9.       For index in No-new-instances:
   10.                Choose a random number
      between 0 and number of neighbours and call it
      nn.
   11.                For attribute in attributes:
   12.                      difference =
      nnarray[nn][attribute] -positive-
      instance[attribute].
   13.                      Gap = generate random
      value between 0 and 1.
   14.                      Synthetic-
      instances.append(positive-instance[attribute]+
      difference * gap)
   15.     Append minority class value to Synthetic-
      instance
End of the function
```

numbering system from 1 to 7 will be used for visualization and for demonstration purposes. Figure 4.2 is the pictorial representation of the claim.

Figure 4.2 shows that the SMOTE instance is generated outside the boundaries of data instance 1 and data instance 2, which means that the claim made in this study about SMOTE method is correct. The larger the space between the value of data instance 1 and data instance 2, the larger the space between the generated data value and the value of data instance 1 and data instance 2. To solve this problem, the difference in the attribute loop of the SMOTE method must be modified to handle the issue of opposite signs.

```
def While_SMT(instances):
    Sythetic_values = []
    for No_New_instances in range(len(instances)):
        row = []
        for attr in range(len(instances[0]) - 1):
            difference = float(instances[0][attr]) - float(instances[1][attr])
            gap = np.random.uniform(0, 1)
            row.append(instances[No_New_instances][attr] + (difference * gap))
        row.append(int(1))
        Sythetic_values.append(row)
    return Sythetic_values
```

Fig. 4.1 SMOTE function

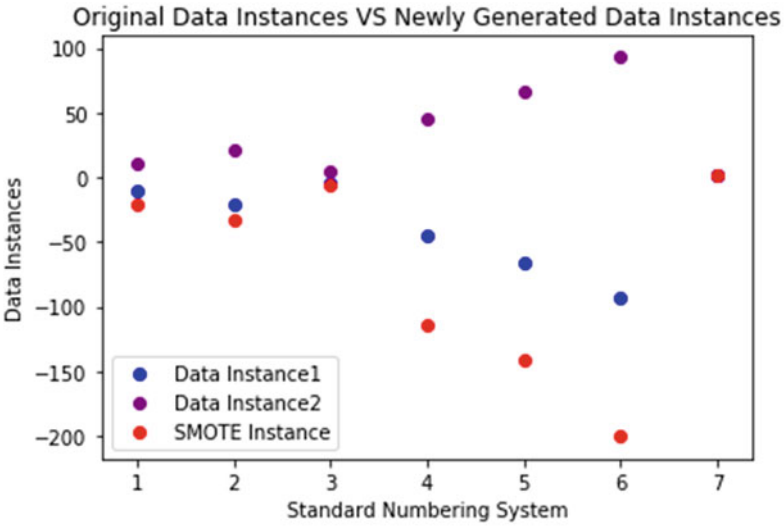

Fig. 4.2 SMOTE demonstration

In this chapter, we propose a random generation of values between the two values of data instance 1 and data instance 2 as a way to ensure that the newly generated values lie within the boundary of data instance 1 and data instance 2. Figure 4.3 shows the modified function of the attribute loop of SMOTE.

The function will use the same values as the above SMOTE function to maintain consistence. Figure 4.4 is the pictorial representation of the claim made by this study that newly generated random values will lie within the boundary of data instance 1 and data instance 2 regardless of the signs.

Figure 4.4 shows that the SMOTE instance is generated within the boundaries of data instance 1 and data instance 2, which means that the claim made in this study about SMOTE method is absolutely correct. Thus, below is the modified algorithm

```
def While_SMT_New(instances):
    Sythetic_values = []
    for No_New_instances in range(len(instances)):
        row = []
        for attr in range(len(instances[0]) - 1):
            row.append(np.random.uniform(instances[0][attr], instances[1][attr]))
        row.append(int(1))
        Sythetic_values.append(row)
    return Sythetic_values
```

Fig. 4.3 SMOTE function (modified)

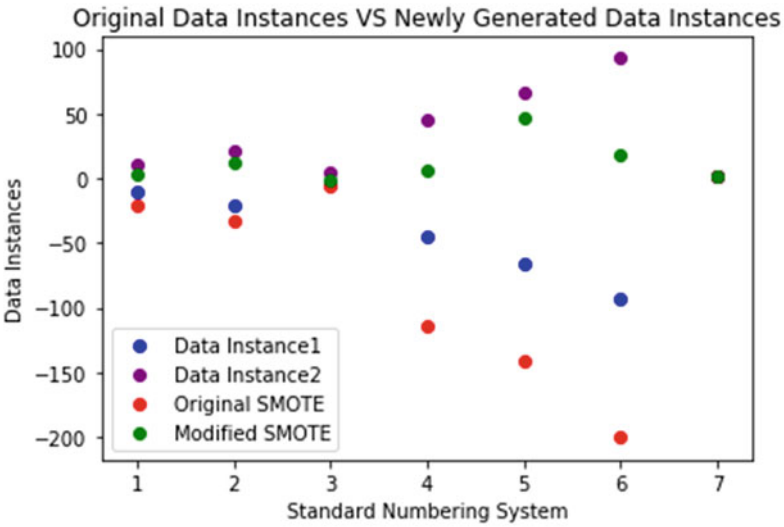

Fig. 4.4 Modified SMOTE demonstration

of SMOTE method that ensures that newly generated data values are generated within the boundaries of data instance 1 and data instance 2.

By generating data instances within the boundaries of data instance 1 and data instance 2, we achieve the objective of the original SMOTE method. Thus, the comparison of SMOTE and safe-level SMOTE algorithms has merit. Below is the definition of the algorithm of safe-level SMOTE.

From the above algorithm, it can be seen that the new data instances that are generated follow the same logic as the SMOTE method above except the fact that the newly generated data values in this case are generated at the safe level of SMOTE. Again, to preserve the primary objective of the SMOTE algorithm, the new data values of safe-level SMOTE must be generated between the data instances at the safe-level of SMOTE. Below is the modified safe-level SMOTE algorithm.

```
Algorithm 2: Modified SMOTE
Input: Minority data observations-T, Size of new
instance in percentage-N,
     Size of the nearest neighbours-K
  Output: I=(N/100)*count(T) - is the newly generated
minority class instances of
                size I - Synthetic-instances.
  Begin
   1. No-of-minority = count(T)
   2. No-new-Instances = I
   3. No-of-attributes = count(T[0] - 1
   4. Minority-instances = T
   5. Synthetic-instances= []
   6.
   7. For positive-instance in positive-instances:
   8.       Compute 5 neighbours of positive-
      instance and call it nnarray.
   9.       For index in No-new-instances:
   10.                 Choose a random number
      between 0 and number of neighbours and call it
      nn.
   11.                 For attribute in attributes:
   12.
      row.append(np.random.uniform(instances[0][attr
      ], instances[1][attr])
   13.     Append minority class value to Synthetic-
      instance
  End of the function
```

The demonstration of how new values of data instances are generated is done from the SMOTE algorithm above. In the data analysis section below, the study compares the performance of the SMOTE and the safe-level SMOTE algorithms.

4.4 Simulation Results

In the data analysis section, the study compares the performance of SMOTE and safe-level SMOTE algorithms, and the modified SMOTE and safe-level SMOTE algorithms by supplying machine learning algorithms with a binary classification dataset that is down sampled by removing duplicate data samples and randomly choose the data observations. The machine learning algorithms chosen in this study

```
Algorithm 3: Safe Level SMOTE
Input: Minority data observations-T, Size of new
instance in percentage-N,
     Size of the nearest neighbours-K
 Output: I=(N/100)*count(T) - is the newly generated
minority class instances of
                size I - Synthetic-instances.
 Begin
  1. No-of-minority = count(T)
  2. No-new-Instances = I
  3. No-of-attributes = count(T[0] - 1
  4. Minority-instances = T
  5. Synthetic-instances= []
  6. For each positive-instance in positive-
     instances:
  7.     Compute 5 neighbours of positive-instance
     and call it K.
  8.     Compute 5 neighbours of positive-instance
     and call it N.
  9.     Count number of positive instances in K
     and call it SLp.
  10.        Count number of positive instances in
     N and call it SLn.
  11.        If SLp != 0:
  12.            SL-ratio = SLp/SLn
  13.        Else:
  14.            SL-ratio = np.inf
  15.        If SL-ratio == np.inf AND SLp ==0:
  16.            Continue.
  17.        Else:
  18.            Feature = []
  19.            For index in
     range(len(np.array(T)[0])):
  20.                If SL-ratio == np.inf AND
     SLp != 0:
  21.                    Gap = 0
  22.                Elseif SL-ratio == 1:
  23.                    Gap =
     np.random.uniform(0,1)
  24.                Elseif SL-ratio > 1:
  25.                    Gap =
     np.random.uniform(0,1 / SL-ratio)
  26.            Elseif SL-ratio < 1:
  27.                Gap = np.random.uniform(1 -
     SL-ratio , 1)
  28.           difference = nnarray[nn][attribute]
     -  positive instance[attribute].
  29.          Gap = generate random value between
     0 and 1.
  30.          Synthetic-instances.append(positive
     - instance[attribute]+ difference * gap)
  31. Append minority class value to Synthetic-
     instance.
End of the function
```

```
  Algorithm 4: Modified Safe Level SMOTE
Input: Minority data observations-T, Size of new
instance in percentage-N,
     Size of the nearest neighbours-K
  Output: I=(N/100)*count(T) - is the newly generated
minority class instances of
                size I - Synthetic-instances.
  Begin
   1. No-of-minority = count(T)
   2. No-new-Instances = I
   3. No-of-attributes = count(T[0] - 1
   4. Minority-instances = T
   5. Synthetic-instances= []
   6. For positive-instance in positive-instances:
   7.     Compute 5 neighbours of positive-instance
      and call it K.
   8.     Compute 5 neighbours of positive-instance
      and call it N.
   9.     Count number of positive instances in K
      and call it SLp.
   10.         Count number of positive instances in
      N and call it SLn.
   11.         If SLp != 0:
   12.             SL-ratio = SLp/SLn
   13.         Else:
   14.             SL-ratio = np.inf
   15.         If SL-ratio == np.inf AND SLp ==0:
   16.             Continue.
   17.         Else:
   18.             Attribute = []
   19.             For index in
      range(len(np.array(T)[0])):
   20.                 If SL-ratio == np.inf AND
      SLp != 0:
   21.                     Gap = 0
   22.                 Elseif SL-ratio == 1:
   23.                     Gap =
      np.random.uniform(0,1)
   24.                 Elseif SL-ratio > 1:
   25.                     Gap =
      np.random.uniform(0,1 / SL-ratio)
   26.                 Elseif SL-ratio < 1:
   27.                     Gap =
      np.random.uniform(1 - SL-ratio , 1)
   28.
      Attribute.append(np.random.uniform(nnarray[nn]
      [attribute],positiveinstances[attribute])
   29.                 Attribute.append[-1] = 1.0
   30.                 Synthetic-
      instance.append(Attribute)
   31.      Append minority class value to
      Synthetic-instance.
  End of the function
```

Table 4.1 Performance comparison of the studied algorithms

	SMOTE	SL SMOTE	SMOTE modified	SL SMOTE modified
Artificial neural network	99.58	99.79	99.58	99.37
Support vector machine	96.46	97.27	96.46	96.86
Naïve Bayesian	100.00	100.00	100.00	100.00
Decision tree	100.00	100.00	100.00	100.00
K-nearest neighbour	97.08	95.38	97.08	94.55

Table 4.2 Confusion matrix: true positives

	SMOTE	SL SMOTE	SMOTE modified	SL SMOTE modified
Artificial neural network	74	66	72	69
Support vector machine	73	64	69	65
Naïve Bayesian	72	67	72	70
Decision tree	74	67	72	70
K-nearest neighbour	69	54	70	70

are Artificial Neural Network, Support Vector Machine, Naïve Bayesian and k-Nearest Neighbour. Table 4.1 shows the performance of SMOTE algorithms in relation to the chosen machine learning algorithms.

The results demonstrate that the oversampling of the minority class plays an important role as shown by the high accuracy of all machine learning models. The oversampling of the minority class was done by generating new instances of the dataset given majority data observations that are down sampled by removing duplicate data samples and randomly choosing the data observations.

The objective of this chapter was to modify the SMOTE and safe-level SMOTE algorithms and compare the performance of original algorithms and modified algorithms. These results show that the safe-level SMOTE algorithm performs better than SMOTE algorithms on credit card fraud data. Furthermore, these results show that the modified SMOTE algorithm performs better than the modified safe-level SMOTE algorithm.

Evaluate Model

The confusion matrix is an evaluation technique that has been used to evaluate the performance of the classification model. The confusion matrix is used to compute the true-positive, false-positive, false-negative and true-negative transactions. The smaller the percentage of false positives and false negatives indicate that the model is actually performing well. Tables 4.2, 4.3, 4.4, and 4.5 show the output of the confusion matrix evaluation technique for each classification model.

The false-negative and false-positive tables show output values which are very small. These small output values indicate that the classification models against oversampling techniques are performing very well.

Table 4.3 Confusion matrix: false positives

	SMOTE	SL SMOTE	SMOTE modified	SL SMOTE modified
Artificial neural network	0	3	0	2
Support vector machine	6	2	2	0
Naïve Bayesian	0	0	0	0
Decision tree	0	0	0	0
K-nearest neighbour	7	12	2	7

Table 4.4 Confusion matrix: false negatives

	SMOTE	SL SMOTE	SMOTE modified	SL SMOTE modified
Artificial neural network	0	1	0	1
Support vector machine	1	3	3	5
Naïve Bayesian	0	0	0	0
Decision tree	0	0	0	0
K-nearest neighbour	5	13	7	2

Table 4.5 Confusion matrix: true negatives

	SMOTE	SL SMOTE	SMOTE modified	SL SMOTE modified
Artificial neural network	381	381	383	378
Support vector machine	375	382	381	380
Naïve Bayesian	381	384	383	380
Decision tree	381	384	383	380
K-nearest neighbour	374	372	376	376

4.5 Results

In conclusion, it can be said that both the SMOTE and safe-level SMOTE are powerful oversampling techniques when they are used with majority data observations that are down sampled by removing duplicate data samples and randomly choosing the data observations. Given that SMOTE and safe-level SMOTE algorithms were modified in this study and the output shows that modified SMOTE performs better than modified safe-level SMOTE; this study therefore concludes that incorrect computation of an algorithm can cloud the algorithm's true computing capabilities.

Acknowledgments I would like thank Prof Ernest Mnkandla for providing insight, expertise and comments that greatly assisted the research. I would also like thank my family: Joyce Noko Dolo, Mahlatse Dolo, Kamogelo Dolo, Tshegofatso Kgoele, Flora Mahlodi Dolo, etc. for their support.

References

1. Gosain, A., & Sardana, S. (2017). Handling class imbalance problem using oversampling techniques: A review. In *2017 international conference on advances in computing, communications and informatics (ICACCI)* (pp. 79–85).
2. Matsuda, K., & Murase, K. (2016). Single-layered complex-valued neural network with SMOTE for imbalanced data classification. In *2016 joint 8th international conference on soft computing and intelligent systems (SCIS) and 17th international symposium on advanced intelligent systems (ISIS)* (pp. 349–354).
3. Pengfei, J., Chunkai, Z., & Zhenyu, H. (2014). A new sampling approach for classification of imbalanced data sets with high density. In *2014 international conference on big data and smart computing (BIGCOMP)* (pp. 217–222).
4. Zeager, M., et al. (2017). Adversarial learning in credit card fraud detection. In *Systems and information engineering design symposium (SIEDS)* (pp. 112–116).
5. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.
6. Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2009). Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 475–482).
7. Meidianingsih, Q., Erfiani, & Sartono, B. (2017). Study of safe-level SMOTE method in unbalanced data classification cases. *International Journal of Scientific and Engineering Research, 8*(5), 1167–1171.
8. TCI. (2015). Driving profitable growth through improved data insights and segmentation practices for the future customer and digital bank. In *Indaba Hotel–Sandton, trade conferences international* (pp. 1–7).