




## Article

# Regularization of Autoencoders for Bank Client Profiling Based on Financial Transactions

Andrey Filchenkov <sup>1,†</sup> , Natalia Khanzhina <sup>1,\*,†</sup> , Arina Tsai <sup>2</sup> and Ivan Smetannikov <sup>1</sup> 

<sup>1</sup> Machine Learning Lab, ITMO University, 49 Kronverksky Pr., 197101 St. Petersburg, Russia; afilchenkov@itmo.ru (A.F.); ismetannikov@itmo.ru (I.S.)

<sup>2</sup> Computer Technologies Department, Formerly ITMO University, 49 Kronverksky Pr., 197101 St. Petersburg, Russia; aaleksandrova@itmo.ru

\* Correspondence: nehazhina@itmo.ru

† These authors contributed equally to this work.

**Abstract:** Predicting if a client is worth giving a loan—credit scoring—is one of the most essential and popular problems in banking. Predictive models for this goal are built on the assumption that there is a dependency between the client’s profile before the loan approval and their future behavior. However, circumstances that cause changes in the client’s behavior may not depend on their will and cannot be predicted by their profile. Such clients may be considered “noisy” as their eventual belonging to the defaulters class results rather from random factors than from some predictable rules. Excluding such clients from the dataset may be helpful in building more accurate predictive models. In this paper, we report on primary results on testing the hypothesis that a client can become a *defaulter* in two scenarios: intentionally and unintentionally. We verify our hypothesis applying data driven regularized classification using an autoencoder to client profiles. To model an intention as a hidden variable, we propose an especially designed regularizer for the autoencoder. The regularizer aims to obtain a representation of defaulters that includes a cluster of *intentional defaulters* and *unintentional defaulters* as outliers. The outliers were detected by our model and excluded from the dataset. This improved the credit scoring model and confirmed our hypothesis.

**Keywords:** clustering; autoencoder; regularization; neural networks; machine learning; credit scoring; transaction profiling; defaulters



**Citation:** Filchenkov, Andrey, Natalia Khanzhina, Arina Tsai, and Ivan Smetannikov. 2021. Regularization of Autoencoders for Bank Client Profiling Based on Financial Transactions. *Risks* 9: 54. <https://doi.org/10.3390/risks9030054>

Received: 2 November 2020

Accepted: 10 March 2021

Published: 17 March 2021

**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

One of the most difficult and highly prioritized banking tasks is assessing the credit-worthiness of clients, which is known as the credit scoring task. The credit score is typically used to predict the loan default risk probability. The absence of the explicit dependency between a client profile and its reliability level makes the task challenging. Currently, all existing solutions are based on various forecasting models, which are usually constructed on prior information about the client. Despite the fact that many banks worldwide spend much on developing novel scoring models and improving the existing ones, this task remains topical, since even a small improvement in the quality of a scoring model can significantly increase the profits [Abellán and Castellano \(2017\)](#).

Scoring models are built on collected datasets containing information about borrowers’ profiles and the target variable, which defines if a client managed to pay off a loan or became a defaulter. Building (training) such models is grounded on an assumption that there is some dependency between borrowers’ behavior before they were given a loan and their behavior afterwards. There are almost unchangeable factors such as psychological traits [Karlan et al. \(2012\)](#); [Lea \(2020\)](#); [Ranyard et al. \(2017\)](#) or even spelling error ratio [Lee and Singh \(2020\)](#) that were shown to affect the probability of borrowers’ default. On the other hand, many unpredictable factors may also affect this probability: global macroeconomic situation, pandemics, death of a relative, sudden disease, or heavy distress. The presence

of such clients in the training data makes the dependency less clear and makes the model harder to train [Silver \(2012\)](#).

In this research, we focus on studying the behavior of clients, who have already been given a loan. We are aware as to which of them were 30 days late in a payment and who were not. Their behavior is represented by their financial transactions performed with a given credit card. Individual spending is a highly informative trace of human behavior that may have a close relationship with social behavioral traits that are important for predicting a personal risk. We assume that investigating such behavior may contribute to understanding, why the loans are not paid off, and improving scoring models.

We hypothesize that there are two groups of clients who default on loans: the first miss payments intentionally, while the second miss them due to some serious reasons. Distinguishing such defaulters is critical for the decision-making process as only the first type of clients should be filtered out while training a predictive model. This paper presents an approach to distinguish defaulters without any knowledge of the ground truth—which of the two categories they belong to—in an unsupervised manner. To solve the task, we propose an autoencoder model with regularization. As the paper reports on work-in-progress, only primary results with a trivial clients' representation are described, which are still promising.

The research questions are the following:

- **if the posterior information about clients' behavior after receiving a loan can be used to improve the credit scoring model based on *prior* information about clients;**
- **if such an improvement can be achieved using transactional data features.**

The contributions of this paper are the following:

1. We hypothesize how defaulters can be distinguished with respect to their transaction profile, formulate a verifiable consequence, and test it.
2. We propose a neural network (NN) model capable of clients profiling based on their transactions that is based on this hypothesis.
3. Finally, we propose the NN-based method for filtering defaulters to detect outliers, who may degrade the scoring model quality.

The rest of the paper has the following structure. We briefly review related works in Section 2. In Section 3, we describe the dataset we use. In Section 4, we present the hypotheses on borrowers' behavior and formulate a verifiable consequence for testing it. The proposed methods for the autoencoder regularization are described in Section 5, followed by the results and discussion in Section 6. The conclusion is given in Section 7.

## 2. Related Work

Optimization of scoring models is a crucially important problem for any creditor. The banking system development and the increase in the number of clients has resulted in the need to automate the solution to this problem. Since the credit scoring problem has been known for a long time and is well researched, there are many already existing solutions based on completely different approaches. State-of-the-art methods of indirect scoring take into account various parameters of the client, such as life situation, credit history, transactional data, etc. Numerous mathematical models have been developed to predict the repayment or to determine clients' behavior patterns. Due to advances in machine learning, this task has received a new round of investigation. One possible statement of the problem, mentioned before, is to predict whether the client makes a loan repayment on-time. This task is solved based on a client profile, which includes not only general information such as gender and age but also financial factors, for example, the payment history. Several publicly available datasets, such as German Credit Data [Asuncion and Newman \(2007\)](#) or Australian Credit Approval [Quinlan \(1987\)](#), became standard benchmarks both for the credit scoring task and the classification in general. There are many papers published on credit scoring, thus we refer only to some of them: [Chen and Huang \(2003\)](#); [Desai et al.](#)

(1996); Hand and Henley (1997); Lee and Chen (2005); Lee et al. (2002); Steenackers and Goovaerts (1989); West (2000).

Previously, Support Vector Machine classifier was the most popular algorithm for credit data processing, for instance, in papers Han et al. (2013); Huang et al. (2007); Li et al. (2006); Van Gestel et al. (2003). Later on, neural networks became extremely popular Khashman (2010); Tsai and Wu (2008); West (2000); Yobas et al. (2000). More recent studies Lessmann et al. (2015) showed the superiority of neural networks and ensembles Tsai and Hung (2014); Wang et al. (2011).

Currently, the research focus has shifted mostly to ensembles Ala'raj and Abbod (2016); Bequé and Lessmann (2017); García et al. (2019); Xia et al. (2017 2018). Most recent surveys can be found in Abellán and Castellano (2017); Dastile et al. (2020) and in Louzada et al. (2016).

Information about clients financial transaction profiles is typically used to detect fraudulent transactions Brown and Pariseau (2009); Gordon (2003); Paulsen et al. (2008); Thakur et al. (2012). Despite the evidence that transactional data could be used for scoring models Zoldi (2013), no academic literature is available, as accessing this type of data may be complicated.

### 3. Dataset

In this research paper, we use a dataset from a local Russian bank. This dataset contains transaction records of 70,000 anonymized clients. Each of the clients was approved by the bank to take out a loan. Thus, the clients in a dataset are known to be creditworthy with respect to the bank decision model.

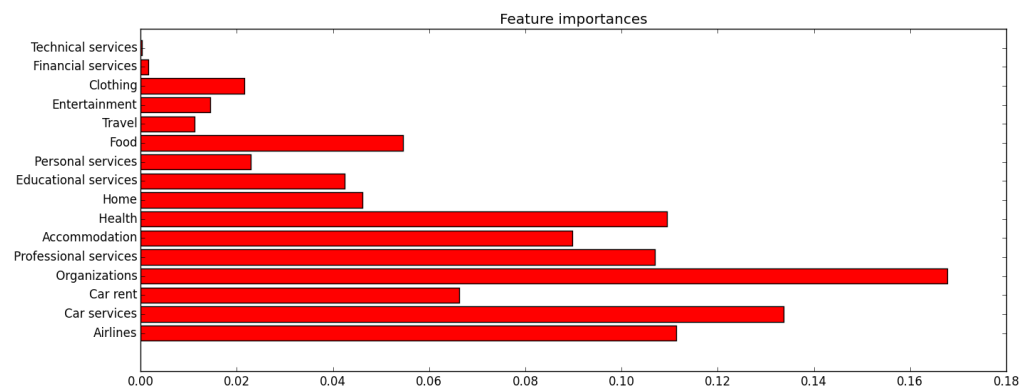
Clients are considered *defaulters* if a loan is not repaid within the 30 day period and are labeled with "1"-s. Clients, who have made the required payments on time, are labeled with "0"-s. A record is a certain client credit card transaction and is described with one of 16 MCCs (Merchant Category Code) the transaction corresponds to, as well as its size (amount of money), date and time, and other specific details.

By the types of client's purchases, location or amount of expenses, a typical portrait, or profile, of a defaulter can be learned to prevent giving loans to people with a similar model of financial behavior.

The dataset has specific features:

1. The data has no information about interest rates, periods, or amounts of loans issued;
2. We are not able to fully assess the client's income or creditworthiness from the data, because the provided transactions do not reflect all of the proceeds to the client's account, including the client's salary.

In this study, we squeeze the transaction records to form aggregated MCC vectors of spendings amount per each client. Thus, our dataset consists of 70,000 weighted vectors of 16 elements and binary labels. The synthesized sample of the dataset can be found here: [http://genome.ifmo.ru/files/papers\\_files/Risks2020/dataset\\_example.csv](http://genome.ifmo.ru/files/papers_files/Risks2020/dataset_example.csv) accessed on 13 March 2021. Each line represents the MCC vector of the client's spendings and the binary flag "defaulter", which is equal to one if the client is a defaulter. We use these vectors as client features. Standard normalization and scaling were applied to the data. The dataset is imbalanced and contains 7000 defaulters only. To handle it, we oversample the minority class in our experiments, which is described in Section 6.2. The dataset feature importance plot with the titles of MCCs is presented in Figure 1.



**Figure 1.** Feature importance plot obtained after applying of Random Forest classifier. The y axes labels are the Merchant Category Codes.

#### 4. A Hypothesis on Defaulters and Verifiable Consequence

First, we introduce a hypothesis on the reasons why borrowers can become defaulters.

**Hypothesis 1.** *Clients can become a defaulter because (1) they took a loan and had no plans to repay it, we call such clients **intentional** defaulters (IDs); or (2) because something happened, which made them unable to pay it off, we call such clients **unintentional** defaulters (UDs).*

Suppose a client who suddenly lost his job, had to cover medical costs, got divorced, or married. All these events are hard to predict in advance by exploring the client's profile when they apply for a loan. However, they significantly affect the client's financial stability in the future and lead to categorizing them as a defaulter. Moreover, unexpected life problems usually do not depend on UD's spending beforehand, therefore, these behavior patterns are not presented in transactions. Hence, the UD's generally behave like *dutiful* borrowers who pay the loan off on time and in full (as opposed to defaulters). It is important to note that this is a simplified and stereotypical description of the two groups. In reality, we suppose that clients show a *tendency* to follow one or another pattern. The motivation behind this tendency is a subject of further research, it may be different from what we described.

Although these two types of clients are indistinguishable for the bank, this difference is very important for predictive models that banks use to decide on a loan approval for the client; unintentional defaulters complicate client analysis.

**Hypothesis 2.** *The behavior of IDs is generally more similar, while the reasons why clients can become UD's may not depend on the personality, but are caused by some various external factors.*

Following this hypothesis, in this study, we perform behavioral profiling via clustering. We assume that the IDs would form a single cluster due to similar behavior patterns, while UD's behave irregularly compared to IDs, thus, they can be referred to as the outliers of the IDs cluster.

**Verifiable Consequence of Hypothesis 2.** Eliminating UD's from the dataset can improve the separability of the defaulting and dutiful borrowers.

In our experiments, the separability of two borrower types is measured based on the classification score. To demonstrate the separability improvement, we suggest comparing the classification scores of the full initial dataset and the filtered dataset. The second one is obtained using client profiling, followed by the detection and elimination of outliers, described in detail in Section 5.

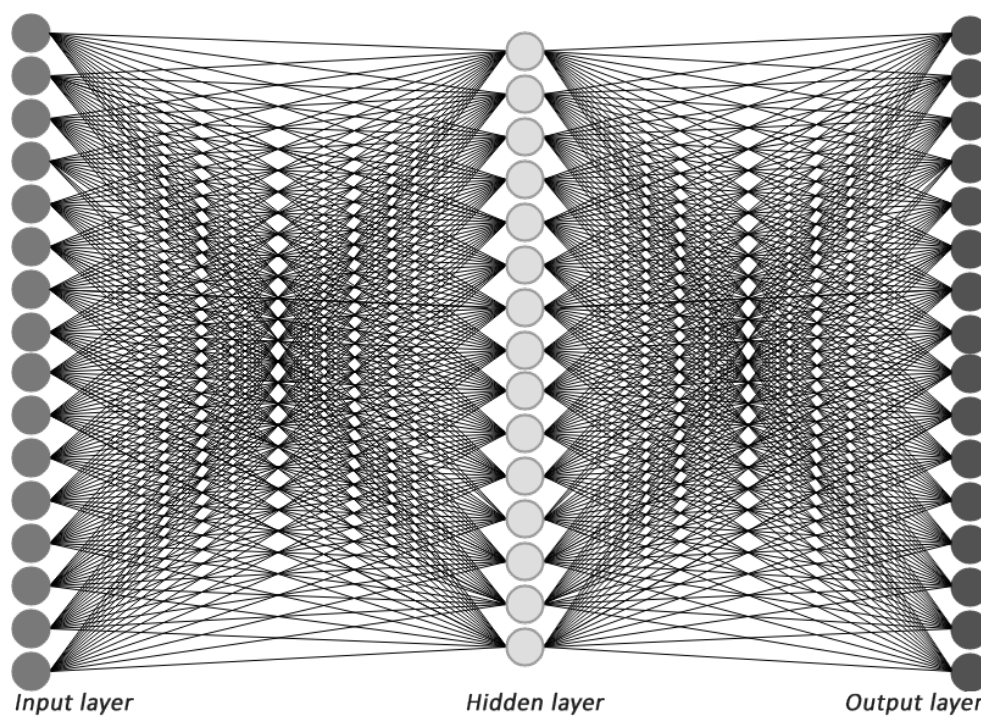
It is worth noting that the hypothesis is not restricted to specific names of behavior patterns, because the ground truth labels for these two groups of clients are essentially unknown.

## 5. Learning the Hypothesis-Driven Representation of Clients with Autoencoder

The described hypotheses involve measuring the similarity of clients, which requires representing clients in some metric space. Despite the clients are described with their transactional profile, it makes sense to learn clients' representation in a unified vector space. As we have no ground truth about which of two groups of defaulters each client belongs to, this is the unsupervised representation learning task. In machine learning, autoencoders are typically used to solve it [Le \(2015\)](#). A crucial benefit of applying autoencoders is that we can guide its training by introducing a special regularization term that will make the data representation to be in a form that we stated in Hypothesis 2.

### 5.1. Autoencoders for Transactional Profiling

To reduce the dimensionality of the data, we use the autoencoder with one hidden layer. The dimension of the hidden layer determines the number of features that we want to extract for describing clients. The autoencoder architecture is presented in [Figure 2](#). In this work, we use the simplest model to have as much control of the learning process as possible.



**Figure 2.** The configure of autoencoder with optimal number of neurons in hidden layer (15). The dimensionality is found using Grid Search over the neurons number from 2 to 15. For more details see [Section 6.4.3](#).

The autoencoder consists of an encoder and a decoder that are trained together: the encoder is trained to approximate a mapping from the input space  $\mathcal{X}$  to a space  $\mathcal{Z}$  of desirable objects representation, and the decoder is trained to approximate the inverse mapping. It worth noting that the mapping is not known in advance, therefore, it can be viewed as searching for the most appropriate space  $\mathcal{Z}$  of lower dimensionality to map the objects to.

The vanilla autoencoder is trained using some reconstruction loss  $\mathcal{L}$  representing how well the autoencoder can restore its input. The learning process can be described as a process of solving the following optimization problem:

$$\mathcal{L}(a_\theta, \mathcal{D}) \rightarrow \min_{\theta \in \Theta}, \quad (1)$$

where  $a_\theta$  is the autoencoder with weights  $\theta$  from some parameter space  $\Theta$  and  $\mathcal{D}$  is a training set. In this work, the loss function used is a mean squared error:

$$\mathcal{L}(a_\theta, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^N \|a_\theta(x_i) - y_i\|^2, \quad (2)$$

where  $\mathcal{D} = \{(x_i, y_i)\}_{i=1 \dots N}$ ,  $x_i$  are data points and  $y_i$  are labels.

We formulated expectations on how the objects behave, and we can guide the autoencoder to find a space in which the objects behave in the expected way allowing restoring input better. This can be done by introducing a regularization term  $\mathcal{R}$  representing how well the learned representation satisfies our expectations. The learning process can now be described as a process of solving the following optimization problem:

$$\mathcal{L}(a_\theta, \mathcal{D}) + \alpha \cdot \mathcal{R}(a_\theta, \mathcal{D}) \rightarrow \min_{\theta \in \Theta}, \quad (3)$$

where  $\alpha$  is a regularization coefficient. We use  $\alpha = 0.1$  to make it of the same order as the value of loss function  $\mathcal{L}$ .  $\mathcal{R}$  is determined by the specific regularization method described in the Sections 5.2 and 5.3.

Since we guide the learning process only by our assumptions and we do not use any labels to train the autoencoders, this scheme is not a subject of overfitting. It is also worth noting that

- Despite the regularization depends only on the output of the hidden layer, we train all the layers with the corresponding loss.
- Weights of the autoencoder are updated iteratively, causing that any relationship between objects in the target representation space can be dramatically changed.

The proposed methods aim to identify similar behavior patterns of clients, stated in Section 4. The regularization can be considered to be data driven. In the remaining subsections we describe three methods to define this regularization term.

### 5.2. Neighbor-Based Minimization Method

To make the cluster of IDs to be well-separable, we need to minimize the distance between points in the cluster and maximize the distance between the cluster points and other points, i.e., outliers. This should result in a single cluster of IDs, while all the outliers should be considered to be UIDs.

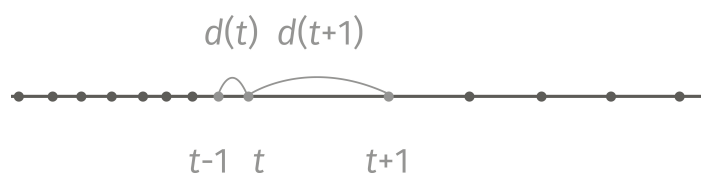
**Neighbor-based minimization method** is designed to solve the task using the density concept. We want the points corresponding to IDs to form a tightly packed cluster. First, for each point  $p$  we calculate the sum of Euclidean distances to its  $k$  nearest neighbors,  $n_p$ . After that, we sort these values:  $n_{(1)} \leq n_{(2)} \leq \dots$ . Finally, we evaluate  $d_i = n_{i+1} - n_i$ . We can guide the autoencoder to obtain such a representation that only one cluster is very tight, while all other points are more distant from each other (or form small clusters). For such a tightly packed cluster,  $n_p$  would be very small and similar to each other, while  $n_q$  for other points would be distinguishably higher, which means that the corresponding values of  $d$  would follow the same pattern.

Here, the following function is considered to be the regularizer for data driven autoencoder training:

$$\mathcal{R}_{NN} = \min\left(\frac{d_t}{d_{t+1}}\right). \quad (4)$$

The minimum of the distance relation function may be achieved on the boundary of the cluster. Therefore, the usage of this function causes the regularizer to maximize the distance between the cluster boundary and outliers and make the cluster tighter. However, we must note that this representation is not robust as it involves reevaluation of such distances after each update of the autoencoder weights. Using this regularization is also not safe as the minimum can be achieved for other points, for instance, for the most distant point and all the other points.

As a result, the representation of the cluster with well-separated outliers can be obtained (Figure 3). Algorithm 1 presents a pseudocode of the method.



**Figure 3.** Illustration of the calculation of the function for neighbor-based minimization method. Here,  $t - 1$  denotes a point within the cluster,  $t + 1$  is an outlier,  $t$  is the cluster boundary point.

---

**Algorithm 1** Algorithm for the neighbor-based minimization method

---

```

for  $i$  in points do
  for  $j$  in points do
     $n_{i,j} \leftarrow$  Euclidean distance ( $i,j$ )
  end for
end for
Sort  $n_i$ 
for  $i$  in points do
  for  $j$  in  $(1 \dots k)$  do
     $n\_sum_i \leftarrow n\_sum_i + n_{i,j}$ 
  end for
end for
Sort  $n\_sum$ 
for  $i$  in points do
   $d_i \leftarrow n\_sum_{i+1} - n\_sum_i$ 
end for
for  $i$  in points do
   $ratio \leftarrow \frac{d_i}{d_{i+1}}$ 
  if  $\mathcal{R}_{NN} > ratio$  then
     $\mathcal{R}_{NN} \leftarrow ratio$ 
  end if
end for

```

---

### 5.3. Barycenter-Based Minimization Methods

Barycenter-based minimization method also aims to form the cluster of *IDs* and separate the *UDs* from it. To do this, the “densest” point should be found first, which we refer to as *barycenter*. That is a point, which distance to *k* nearest neighbors is minimal. This point can be considered to be the center of the cluster. Here, the regularizer is expected to bring the cluster point closer to its center and bring outliers away. For this purpose, we evaluate Euclidean distances from each point to its *k* nearest neighbors and sum these distances. The point with the minimum sum value is selected as the barycenter *b*. Then, for the barycenter *b* the Euclidean distances to every other point *p*,  $\rho(b, p)$  are evaluated. We sort them and evaluate differences to make the series more robust:  $\rho_{(1)} \leq \rho_{(2)} \leq \dots$ ,  $r_t = \rho_{(t+1)} - \rho_{(t)}$ . The minimization function is defined in the same way with the previous minimization method:

$$\mathcal{R}_{Bc} = \min\left(\frac{r_t}{r_{t+1}}\right). \quad (5)$$

The minimum is reached on the cluster boundary, determining its radius. Thus, the ratio of the cluster radius and the closest to the cluster outlier is minimized. In our experiments, we make a constraint on the cluster size, restricting its infimum and supremum size to make the cluster containing 20–50% of all the points.

This method is a more robust than the first one because its values depend only on distances to a single point, which do not change that drastically after weights updates. However, the stability may be affected when after weights update a new point is chosen as a new barycenter, which eliminates effects of training the representation with respect to our expectation. There can be an issue when another point is becoming a boundary point, so the learning progress is also a bit eliminated.

To overcome the described problems, we introduce **stabilized barycenter-based method**, aimed at increasing the stability of the cluster radius using an improved regularizer function. In this modification, the cluster center is fixed over the forward passes of the autoencoder making the radius dependent on it. In addition, the minimization of the radius ratio here depends on the previous value of the cluster radius, which is provided by the multiplier. Thus, the regularizer is represented as a hysteresis function:

$$\mathcal{R}_{SBc} = \min\left(\frac{r_t}{r_{t+1}} \cdot e^{|\ln\left(\frac{r_b}{r_t}\right)|}\right), \quad (6)$$

where  $r_b$  is the cluster radius at the previous training step. If the difference between radii is small, this multiplier value is close to 1, otherwise, it is greater than 1, thereby increasing the product value. The method pseudo-code is presented below (Algorithm 2):



**Algorithm 2** Algorithm for the stabilized barycenter-based method

---

```

for  $i$  in  $points$  do
  for  $j$  in  $points$  do
     $\rho_{i,j} \leftarrow$  Euclidean distance ( $i,j$ )
  end for
end for
Sort  $\rho_i$ 
for  $i$  in  $points$  do
  for  $j$  in  $(1 \dots k)$  do
     $\rho\_sum_i \leftarrow (\rho\_sum_i[0] + \rho_{i,j}, i)$ 
  end for
end for
Sort  $\rho\_sum$ 
 $b \leftarrow \rho\_sum[0][0]$ 
 $b\_index \leftarrow \rho\_sum[0][1]$ 
for  $i$  in  $points$  do
   $r_i \leftarrow \rho_{b\_index,i+1} - \rho_{b\_index,i}$ 
end for
for  $i$  in  $points$  do
   $ratio \leftarrow \frac{r_i}{r_{i+1}} \cdot \exp\left|\ln \frac{r_b}{r_i}\right|$ 
  if  $\mathcal{R}_{SBc} > ratio$  then
     $\mathcal{R}_{SBc} \leftarrow ratio$ 
     $r_b \leftarrow point_i$ 
  end if
end for

```

---

**6. Experiments and Results****6.1. Experimental Setup**

In this research paper, the following experimental pipeline is used:

1. First, various predictive models for credit scoring are trained and evaluated on the full initial dataset. The best classifier is chosen as the baseline.
2. The autoencoder models are trained on the full dataset (as opposed to the filtered one). Three proposed regularization methods result in three trained models.
3. After that, these models are used for defaulters filtering (i.e., the forward pass is performed for them only) based on the encoder output.
4. To evaluate the encoder dimensionality reduction impact, principal component analysis (PCA) is applied to the full dataset.
5. To compare the proposed filtering approach, the dataset random filtering is performed as well. The filtered datasets should be identically distributed.
6. Finally, the obtained datasets are used to train and evaluate the best baseline classifier for further results comparison.

In our experiments, the 70/30 as train/test ratio and the 5-fold cross-validation is used.

**6.2. Performance Evaluation and Implementation Details**

The evaluation is based on the *weighted F<sub>1</sub>-score* and the *AUC score* as metrics, which are defined as follows.

$$F_1 = 2 \cdot \frac{\textit{Precision} \cdot \textit{Recall}}{\textit{Precision} + \textit{Recall}} \quad (7)$$

where *Precision* is the percentage of correctly classified objects relative to those that the model assigned to this class, *Recall* refers to the percentage of correctly recognized objects relative to all objects of the same class. We use weighted  $F_1$ -score, which weights the score of each class by the number of samples in that class.

The area under the curve, or *AUC*, is an aggregated characteristic of classification score and is based on the ROC (Receiver Operating Characteristic) curve. In turn, ROC plots the dependence between True Positive Rate (TRP) and False Positive Rate (FPR). AUC varies between 0 and 1. The higher the AUC value, the better the predictive model.

As the dataset is highly imbalanced, we first perform its balancing using the imblearn library ([https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over\\_sampling.SMOTE.html](https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html) accessed on 13 March 2021). We use the SMOTE oversampling method, which allows preserving the data distribution. This method synthesized new samples of the minority class based on the  $k$  nearest neighbors of a randomly selected instance of the minority class. A new point is taken between this instance and its randomly selected neighbor. We use  $k = 5$ . All the reported results are obtained on the balanced datasets.

All the methods are implemented with TensorFlow2/Keras and scikit-learn Python libraries. For experiments, we used GeForce GTX 1080 Ti GPU.

### 6.3. Classification Methods

We use the following algorithms to train predictive models Bishop (2006); Friedman (2001); Boswell (2002); Hastie et al. (2009); Mitchell (1997).

1. Logistic regression (LR). A simple classification technique is based on the logistic function to model a dependence.
2. Decision trees (DT). The idea is to recursively split a feature space with a feature value until some criterion is reached (e.g., a tree leaf has a minimum number of target classes).
3. Feedforward neural network (NN). This is the simplest type of artificial neural network that includes fully connected layers.
4. K-nearest neighbors (kNN). A metric classification technique, which assigns a class to an object based on its  $k$  nearest neighbors classes.
5. Random forest (RF). An ensemble method, in which several independent models make predictions; after that, the final prediction is formed by voting in case of the classification problem.
6. Gradient boosting (GB). This is an ensemble method, which minimizes the training error of classifier linear composition based on gradient descent.

### 6.4. Results

Following the experimental pipeline, we first classify the full dataset to obtain the baseline credit scoring model. The results are presented in Table 1. The optimal hyperparameters of the classifiers are found using *Grid Search*, which is a full parameter enumeration method.

The best result is provided by *Gradient Boosting Machine (GB)*.

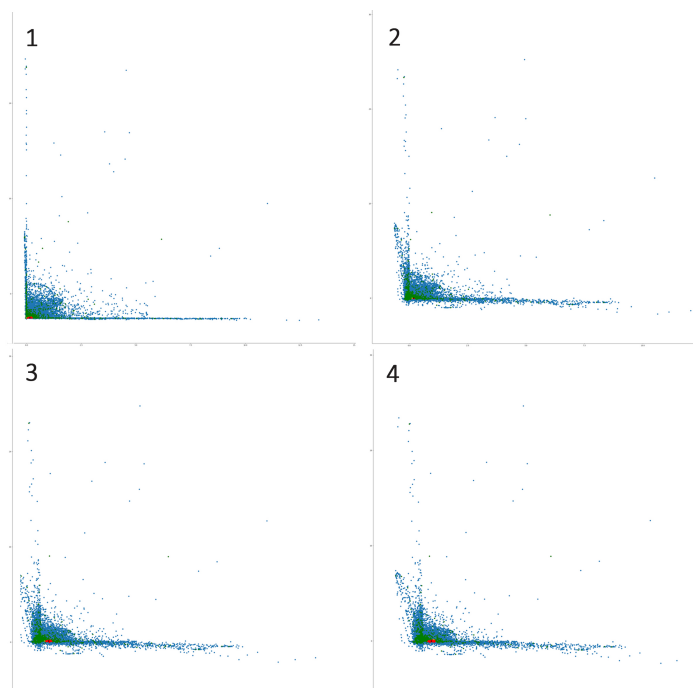
**Table 1.** Baseline classification models and their performance. The metrics are F1-measure and AUC.

| Model                  | F1                 | AUC         |
|------------------------|--------------------|-------------|
| Log. regression (LR)   | 0.56 ± 0.04        | 0.59        |
| Decision tree (DT)     | 0.59 ± 0.04        | 0.63        |
| Neural network (NN)    | 0.59 ± 0.02        | 0.65        |
| kNN                    | 0.79 ± 0.01        | 0.80        |
| Random forest (RF)     | 0.73 ± 0.01        | 0.73        |
| Gradient boosting (GB) | <b>0.83 ± 0.01</b> | <b>0.83</b> |

To demonstrate the impact of the methods proposed, we need to compare them to the baseline scoring model as follows. We train the three autoencoder models with different regularizers. For each model, the encoded features are received on the hidden layer of the autoencoder. These features are clustered with a certain method, and the outliers are identified. As a result, the outliers are filtered with the trained model, obtaining a new dataset. On the new datasets, the best classifier (*GB*) is trained, and the results of new scoring models are compared with the baseline. With this approach we test the verifiable consequence of Hypothesis 2.

#### 6.4.1. Neighbor-Based Minimization Method Results

The *neighbor-based minimization* method has not clustered the *IDs* as expected. Firstly, as this method does not fix the center of the cluster over the training steps, the center may change, resulting in the cluster being “tightened” to different centers at every training step. This instability does not allow obtaining the tight *IDs* cluster, which affects the model training. Secondly, with the cluster center unfixed, the minimization according to Equation (4) causes the method to find small clusters and consider the distance ratio with respect to them, instead of forming a single cluster. The visualized process of clustering training is represented in Figure 4.

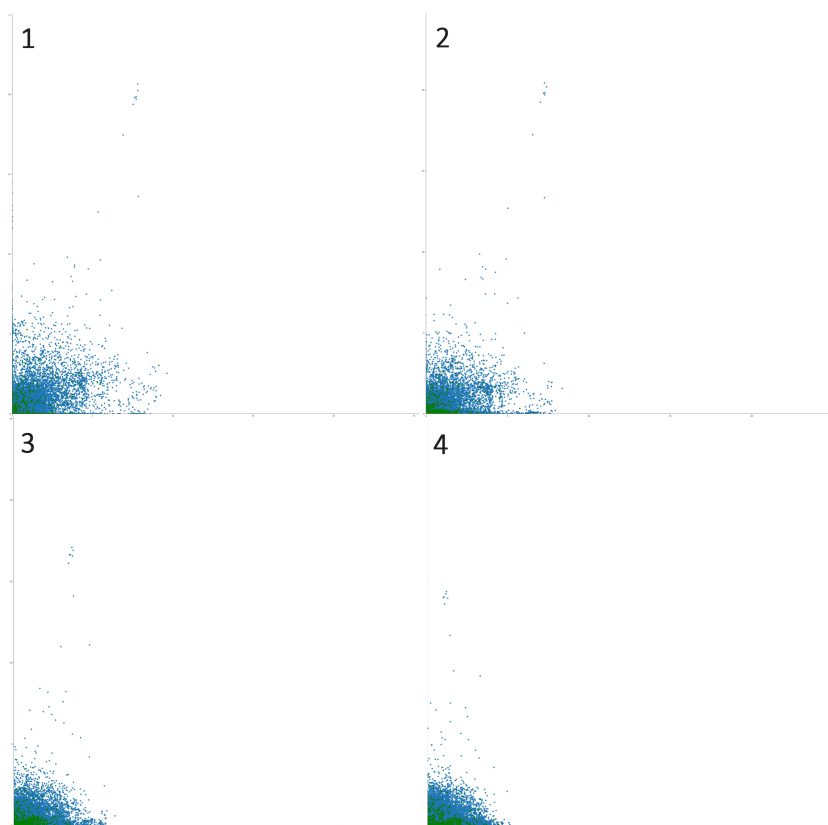


**Figure 4.** The examples of clustering using the neighbor-based minimization method obtained on four consequent encoder forward passes. Green color points are the *IDs*, blue color points are the *UDs*.

The model is not able to detect the outliers. Hence, it is excluded from further comparison with the baseline.

#### 6.4.2. Barycenter-Based Minimization Method Results

The *barycenter-based minimization* method obtains the well-clustered *IDs* point representation using the fixed cluster center. This fixing allowed to stabilize the optimization process using the regularizer from Equation (6). In our experiments, we test both method modifications: with the stabilizing multiplier and without it. Figure 5 demonstrates the clustering results during the model training. Only this method (original and stabilized) participates in the further comparison.

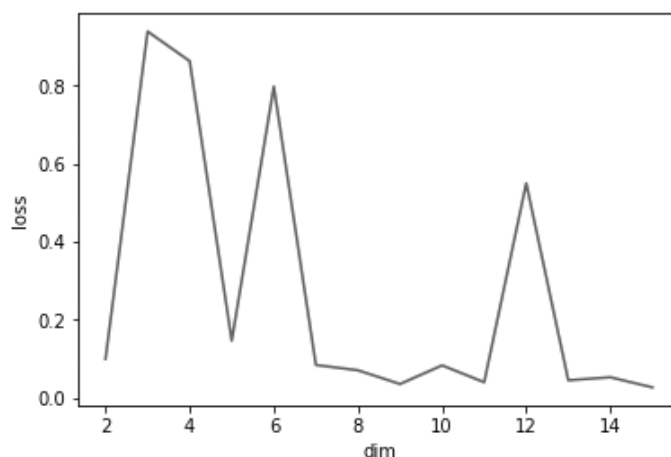


**Figure 5.** The examples of clustering using the barycenter-based minimization method obtained on four consequent encoder forward passes. Green color points are the *IDs*, blue color points are the *UDs*.

#### 6.4.3. Results Comparison and Discussion

For the comparison of the results, we train *GB* on the data with the three different feature sets: (1) with the original 16 features, (2) with features obtained by the encoder, and (3) with the features obtained by the *PCA* algorithm. In addition, to ensure that the obtained result does not depend on the distribution of the dataset (the ratio of defaulting and dutiful borrowers), we conduct experiments with random filtering the same number of defaulters from the full dataset as in the dataset obtained by the proposed filtering method. Thus, we train *GB* on datasets of three different sizes: the full dataset, the dataset filtered randomly, and the dataset filtered using the proposed method.

To choose the dimensionality of the encoder, we conducted the experiments with the full *Grid Search* over the neurons number from 2 to 15 (for the results see Figure 6). The best result was achieved using 15 neurons in the encoder output layer. Hence, the *PCA* output dimensionality was also set to 15 components.



**Figure 6.** The dependence of the autoencoder loss function value from the dimension of encoder after training on the full dataset. The *y* axis represents the value of the loss function, the *x* axis represents the number of neurons in the hidden layer. The best result is achieved with the number of neurons equal to 15.

The cross-validated results of GB-based credit scoring applied to eight datasets are presented in Tables 2–4. In this study, the PCA method was not applied to the filtered dataset obtained by the method proposed, because this would change the data distribution. The two ranges of the cluster size were tested for the *barycenter-based minimization* method: from 20% to 80%, and from 50% to 89% of initial points number. Here, the upper limit ensures that the cluster does not include all the points. The two configurations of the barycenter-based minimization method were tested: *with* the stabilizer and *without* the stabilizer.

Our experiments have demonstrated that the proposed data-driven regularization method is superior to the baseline model on all the datasets tested. The best result is achieved using the stabilized barycenter-based method, which proves the importance of taking into account the cluster size from the previous training step. The developed method improves the solution of the credit scoring problem and provides better performance than models without clients transaction profiling.

**Table 2.** Results of GB-based classification of the datasets with different filtering methods and different feature sets. The stabilized barycenter-based (SBc) method limits the cluster size to be in range 20–80%. The *ID*s obtained cluster size is 1709 with 1341 of outliers (*UD*s).

|                       | Original Dim | AE Hidden Layer Output | PCA Output    |
|-----------------------|--------------|------------------------|---------------|
| Full DS               | <b>0.834</b> | 0.7385                 | 0.7364        |
| Random filtering DS   | 0.7380       | 0.7378                 | <b>0.7378</b> |
| SBc regularization DS | 0.833        | <b>0.850</b>           | -             |

**Table 3.** Results of GB-based classification of the datasets with different filtering methods and different feature sets. The barycenter-based minimization (Bc) *without* a stabilizer method limits the cluster size to be in range 50–89%. The *ID*s obtained cluster size is 1709 with 831 of outliers (*UD*s).

|                      | Original Dim | AE Hidden Layer Output | PCA Output    |
|----------------------|--------------|------------------------|---------------|
| Full DS              | <b>0.834</b> | 0.7375                 | <b>0.7364</b> |
| Random filtering DS  | 0.832        | 0.7357                 | 0.7347        |
| Bc regularization DS | 0.831        | <b>0.848</b>           | -             |

**Table 4.** Results of GB-based classification of the datasets with different filtering methods and different feature sets. The stabilized barycenter-based (SBc) method limits the cluster size to be in range 50–89%. The *ID*s obtained cluster size is 1709 with 730 of outliers (*UID*s).

|                       | Original Dim | AE Hidden Layer Output | PCA Output    |
|-----------------------|--------------|------------------------|---------------|
| Full DS               | <b>0.834</b> | 0.7315                 | <b>0.7364</b> |
| Random filtering DS   | 0.83         | 0.7336                 | 0.7320        |
| SBc regularization DS | 0.833        | <b>0.851</b>           | -             |

As we can see, filtering out those who were considered to be outliers by the autoencoder has improved the predictive quality of the scoring model compared with the original dataset and with the dataset of the same size with randomly filtered objects. As the autoencoder regularization is based only on the defaulters, its representation is not overfitted towards discrimination of defaulters and non defaulters. We therefore conclude that **the verifiable consequence is confirmed**. This is a **good evidence confirming Hypothesis 2**, which in its turn, is **evidence confirming Hypothesis 1**.

However, we must notice that stronger evidence should be achieved to confirm Hypothesis 2 and, especially, Hypothesis 1. There may be other reasons why filtering out objects in this way improves the predictive accuracy. Training a representation, in which we leave only a tightly packed cluster of objects of one class may ease the process of separating these objects from objects of another class.

## 7. Discussion and Conclusions

In this paper, we proposed a competitive methodology for the problem of binary credit scoring based on borrowers' financial transactions. We formulated the hypothesis for the profiling of bank's clients, who took a loan, using their transactional data. It was assumed that a client can become a defaulter either intentionally or unintentionally, and identifying unintentional defaulters as outliers for further elimination can enhance the credit scoring model.

To find the optimal client representation, we attempted to reduce the feature space dimensionality using autoencoders. As for the profiling approach, we developed three methods of autoencoder regularization. The proposed regularizers are used to cluster clients in the output space of the encoder and are aimed at obtaining a tightly packed cluster of intentional defaulters with the outlied unintentional defaulters.

We proved that the proposed profiling methods efficiently filter clients, which allows the classification model for credit scoring to generalize better. The scoring model applied after the proposed filtering method outperformed the same model applied to the identically distributed random filtered dataset. Additionally, our experiments showed that the proposed encoder model is superior to the *PCA* method with the same number of components. The obtained results prove the hypotheses stated in this research and opens a new direction in approaches to the problem of credit scoring.

In our study, we used only one hidden layer of the autoencoder, leaving the deeper models for further research. We also used the reduced, or squeezed, client representation, disregarding the transactions as sequences. However, considering temporal connections may have a significant impact on the task solution. As a direction for future work, recurrent models, namely LSTMs Gers et al. (2002), as well as Transformers Vaswani et al. (2017) should be investigated.

**Author Contributions:** Conceptualization, A.F. and N.K.; methodology, N.K.; software, A.T.; validation, N.K. and A.T.; formal analysis, A.F.; investigation, A.T. and N.K.; resources, N.K.; data curation, A.F.; writing—original draft preparation, A.T.; writing—review and editing, A.F., N.K. and I.S.; visualization, A.T. and N.K.; supervision, A.F.; project administration, N.K.; funding acquisition, A.F., N.K. and I.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is financially supported by National Center for Cognitive Research of ITMO University and the Russian Science Foundation, Agreement 17-71-30029.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** The authors would like to thank Anton Belyy, Arip Asadulaev, Klavdia Bochenina, and Alexander Boukhanovsky for useful conversations, Tatyana Polevaya, and Inna Anokhina for useful comments.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

- Abellán, Joaquín, and Javier G. Castellano. 2017. A comparative study on base classifiers in ensemble methods for credit scoring. *Expert Systems with Applications* 73: 1–10. [CrossRef]
- Ala'raj, Maher, and Maysam F. Abbod. 2016. A new hybrid ensemble credit scoring model based on classifiers consensus system approach. *Expert Systems with Applications* 64: 36–55. [CrossRef]
- Asuncion, Arthur, and David Newman. 2007. Uci Machine Learning Repository. Available online: <https://archive.ics.uci.edu/ml/index.php> (accessed on 13 March 2021).
- Bequé, Artem, and Stefan Lessmann. 2017. Extreme learning machines for credit scoring: An empirical evaluation. *Expert Systems with Applications* 86: 42–53. [CrossRef]
- Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Berlin: Springer.
- Boswell, Dustin. 2002. *Introduction to Support Vector Machines*. San Diego: Department of Computer Science and Engineering University of California San Diego. Available online: <http://pzs.dstu.dp.ua/DataMining/svm/bibl/IntroToSVM.pdf> (accessed on 1 November 2020).
- Brown, Kerry D., and David Kevin Pariseau. 2009. Financial Transactions with Dynamic Card Verification Values. U.S. Patent No. 7,584,153, September 1.
- Chen, Mu-Chen, and Shih-Hsien Huang. 2003. Credit scoring and rejected instances reassigning through evolutionary computation techniques. *Expert Systems with Applications* 24: 433–41. [CrossRef]
- Dastile, Xolani, Turgay Celik, and Moshe Potsane. 2020. Statistical and machine learning models in credit scoring: A systematic literature survey. *Applied Soft Computing* 91: 106263. [CrossRef]
- Desai, Vijay S., Jonathan N. Crook, and George A. Overstreet Jr. 1996. A comparison of neural networks and linear scoring models in the credit union environment. *European Journal of Operational Research* 95: 24–37. [CrossRef]
- Friedman, Jerome H. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 2001: 1189–232. [CrossRef]
- García, Vicente, Ana I. Marqués, and J. Salvador Sánchez. 2019. Exploring the synergetic effects of sample types on the performance of ensembles for credit risk and corporate bankruptcy prediction. *Information Fusion* 47: 88–101. [CrossRef]
- Gers, Felix A, Nicol N Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research* 3: 115–43.
- Gordon, Goren. 2003. Method and System of Data Analysis for the Detection of Fraudulent Financial Transactions. U.S. Patent Application No. 10/152,169, May 22.
- Han, Lu, Liyan Han, and Hongwei Zhao. 2013. Orthogonal support vector machine for credit scoring. *Engineering Applications of Artificial Intelligence* 26: 848–62. [CrossRef]
- Hand, David J., and William E. Henley. 1997. Statistical classification methods in consumer credit scoring: A review. *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 160: 523–41. [CrossRef]
- Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. New York: Springer.
- Huang, Cheng-Lung, Mu-Chen Chen, and Chieh-Jen Wang. 2007. Credit scoring with a data mining approach based on support vector machines. *Expert Systems with Applications* 33: 847–56. [CrossRef]
- Karlan, Dean, Sendhil Mullainathan, and Omar Robles. 2012. Measuring personality traits and predicting loan default with experiments and surveys. *Banking the World: Empirical Foundations of Financial Inclusion* 2012: 393.
- Khashman, Adnan. 2010. Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes. *Expert Systems with Applications* 37: 6233–39. [CrossRef]
- Le, Quoc V. 2015. A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks. *Google Brain* 2015: 1–20.
- Lea, Stephen EG. 2020. Debt and overindebtedness: Psychological evidence and its policy implications. *Social Issues and Policy Review* 15: 1–34.
- Lee, Tian-Shyug, and I-Fei Chen. 2005. A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines. *Expert Systems with applications* 28: 743–52. [CrossRef]

- Lee, Michelle Seng Ah, and Jatinder Singh. 2020. Spelling Errors and Non-Standard Language in Peer-to-Peer Loan Applications and the Borrower's Probability of Default. SSRN 3609834. Available online: <https://clck.ru/ThSLQ> (accessed on 1 November 2020).
- Lee, Tian-Shyug, Chih-Chou Chiu, Chi-Jie Lu, and I-Fei Chen. 2002. Credit scoring using the hybrid neural discriminant technique. *Expert Systems with Applications* 23: 245–54. [CrossRef]
- Lessmann, Stefan, Bart Baesens, Hsin-Vonn Seow, and Lyn C Thomas. 2015. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research* 247: 124–36. [CrossRef]
- Li, Sheng-Tun, Weissor Shiue, and Meng-Huah Huang. 2006. The evaluation of consumer loans using support vector machines. *Expert Systems with Applications* 30: 772–82. [CrossRef]
- Louzada, Francisco, Anderson Ara, and Guilherme B. Fernandes. 2016. Classification methods applied to credit scoring: Systematic review and overall comparison. *Surveys in Operations Research and Management Science* 21: 174. [CrossRef]
- Mitchell, Tom M. 1997. *Machine Learning 1*. New York: McGraw-Hill Education.
- Paulsen, Kobus, Ian Hughes, and Mark Holland. 2008. Systems and Methods for Identifying Potentially Fraudulent Financial Transactions and Compulsive Spending Behavior. U.S. Patent Application No. 11/609,785, February 14.
- Quinlan, J. Ross. 1987. Simplifying decision trees. *International Journal of Man-Machine Studies* 27: 221–34. [CrossRef]
- Ranyard, R., Sandie Mchugh, and Simon Mcnair. 2017. The psychology of borrowing and over-indebtedness. *Economic Psychology* 2380: 222–38.
- Silver, Nate. 2012. *The Signal and the Noise: The Art and Science of Prediction*. London: Penguin UK.
- Steenackers, A., and Marc Goovaerts. 1989. A credit scoring model for personal loans. *Insurance: Mathematics & Economics* 8: 1–4.
- Thakur, Aman, Jeffrey A. Hughes, and Sanjay Suri. 2012. Indicating Irregularities in Online Financial Transactions. U.S. Patent No. 8,290,838, October 16.
- Tsai, Chih-Fong, and Chihli Hung. 2014. Modeling credit scoring using neural network ensembles. *Kybernetes* 43: 114–23. [CrossRef]
- Tsai, Chih-Fong, and Jhen-Wei Wu. 2008. Using neural network ensembles for bankruptcy prediction and credit scoring. *Expert Systems with Applications* 34: 2639–49. [CrossRef]
- Van Gestel, Ir Tony, Bart Baesens, Ir Joao Garcia, and Peter Van Dijcke. 2003. A support vector machine approach to credit scoring. In *Forum Financier-Revue Bancaire et Financieraire Bank en Financiewezen*, pp. 73–82. Available online: <https://clck.ru/ThSPm> (accessed on 1 November 2020).
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Paper presented at the Advances in Neural Information Processing Systems, Long Beach, CA, USA, December 4–9; pp. 5998–6008.
- Wang, Gang, Jinxing Hao, Jian Ma, and Hongbing Jiang. 2011. A comparative assessment of ensemble learning for credit scoring. *Expert Systems with Applications* 38: 223–30. [CrossRef]
- West, David. 2000. Neural network credit scoring models. *Computers & Operations Research* 27: 1131–52.
- Xia, Yufei, Chuanzhe Liu, YuYing Li, and Nan Liu. 2017. A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring. *Expert Systems with Applications* 78: 225–41. [CrossRef]
- Xia, Yufei, Chuanzhe Liu, YuYing Li, and Nan Liu. 2018. A novel heterogeneous ensemble credit scoring model based on bstacking approach. *Expert Systems with Applications* 93: 182–99. [CrossRef]
- Yobas, Mumine B., Jonathan N. Crook, and Peter Ross. 2000. Credit scoring using neural and evolutionary techniques. *IMA Journal of Management Mathematics* 11: 111–25. [CrossRef]
- Zoldi, Scott. 2013. Big Data Developments in Transaction Analytics. Available online: <https://www.business-school.ed.ac.uk/crc/wp-content/uploads/sites/55/2017/02/Big-Data-Developments-in-Transaction-Analytics-Scott-Zoldi.pdf> (accessed on 1 November 2020).