Predictive Modeling and Segmentation of Credit Card Default Risk

Roger Bukuru (BKRROG001)

University of Cape Town
Multivariate Statistics (STA50697Z)
Final Report

Abstract

Banks are essential to the financial services industry because they provide their clients with credit and loans. These financial tools are essential for empowering customers financially as well as allowing banks to earn interest and generate additional revenue. However, the granting of credit and loans carries with it an inherent default risk, meaning that borrowers might not fulfil their repayment commitments.

With an emphasis on a Taiwanese credit default dataset, this study aims to tackle the problem of credit risk management, specifically as it relates to credit card facilities. Finding the best effective dimensionality reduction method to use with k-means clustering among Principal Component Analysis (PCA), Autoencoders (AE), and Variational Autoencoders (VAE) is the main goal of our work.

There are two questions in the research. Its initial goal is to determine which dimension reduction method best facilitates the creation of an elementary credit default risk segmentation model. This entails investigating the ways in which a customer's payment history, bill statement amounts, prior payment amounts, and demographic and financial behaviour traits—such as age, gender, education, marital status, and marriage status—can affect the probability of a credit default. The second portion of the study topic deals with the viability of creating a trustworthy default prediction model based on the best customer segmentation model that has been determined after a segmentation model has been established. Support Vector Machines (SVMs) will be utilized in order to build the predictive model.

In order to improve customer segmentation, dimensionality reduction techniques such as PCA, AE, and VAE are being used to extract the key characteristics that affect credit default risk. This segmentation then forms the basis for predictive modelling, which allows for a more sophisticated evaluation of credit default risk. The repository of this study is available at https://github.com/rogerbukuru/Predictive-Modeling-and-Segmentation-of-Credit-Card-Default-Risk.git

Keywords: credit card; default; bank; risk profiles; customer-segmentation; support vector machines, principal component analysis; autoencoders; variational autoencoders;

1 Introduction

Credit card lending is a widely researched subject, our focus in this study focuses more particularly on credit risk segmentation and prediction. An important risk management technique for banks and credit institutions in the hopes of reducing default risk with such financial instruments[8].

[4] illustrates that a number of statistical methods have been applied to develop reliable credit risk prediction models. In their research on credit card defaults, [4] implement feature selection and extraction techniques such as Principal Component Analysis(PCA) and Linear Discriminate Analysis (LDA). They proceeded to apply classification methods including K-nearest neighbour (KNN), Naive Bayesian (NB), Decision Tree (DT), and Support Vector Machines (SVM) to analyze credit card client defaults, identifying the most effective technique for assessing credit card default risk. Their findings indicated that SVM when combined with feature selection and extraction methods, achieved the highest predictive accuracy.

[8] went with the approach of placing customers within segments that represent various default risk profiles. This was conducted by using clustering techniques such as Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Self-Organzaing Maps (SOMs). For credit default prediction they utilized a tree-based method, Gradient Boosting. Their segmentation results were unsatisfactory due to issues such as too few feature variables, with the credit card default data they utilized. On the other hand, their prediction model produced high enough certainty that the bank for which the study was conducted could utilize it in its functions.

[11] point out that from the perspective of risk control, estimating the probability of default will be more meaningful than classifying customers into binary results– risky and non-risky. This is however highlighted to be a difficult task given credit issuers do not have a real probability of default. This is however attempted with 6 classification techniques including a k-nearest neighbour, logistic regression, discriminant analysis, naïve-Bayes, neural networks, and classification trees. Artificial neural networks proved to be the most reliable both in terms of classification and predictive accuracy regarding credit risk. Our main objective will be to develop an elementary credit risk segmen-

tation model utilizing k-means clustering in conjunction with PCA, Autoencoders, and Variational Autoencoders. Predictive modelling will be evaluated once these segmentations have been accomplished through the integration of segmentation strategies and Support Vector Machine classification. In order to maintain clarity and conciseness while addressing the research inquiries, we decided to include all data preprocessing techniques utilized in the appendix for perusal. All analyses were conducted subsequent to the highlighted preprocessing steps.

2 Exploratory Data Analysis

2.1 Data Description

The data we will be performing our analysis consists of 30,000 observations that represent default payments of distinct credit card holders from a bank (a cash and credit card issuer) in Taiwan from April 2005 to September 2005[11]. The dataset utilizes a binary variable, default payment (Yes =1, No=0), as the response variable.

The data consists of the following 23 predictor variables:

The initial set of variables comprises details regarding the client's personal information:

- Limit Balance: A customer credit limit
- Sex: (1 - Male; 2 - Female)
- Education: 1 = graduate school; 2 = university; 3 = high school; 4 = others
- Marriage: 1 - Married; 2 - Single; 3 - Other
- Age: Customer age

The subsequent attributes provide details on the delay in past payments corresponding to specific months:

- Pay0 - Pay11: History of past payment. We tracked the past monthly payment records (from April to September 2005) as follows: Pay0 = the repayment status in September 2005; Pay2 = the repayment status in August 2005; . . .; Pay6 = the repayment status in April 2005. The measurement scale for the repayment status is -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.

The following variables focus on information concerning the bill statement amount (that is, a monthly statement issued by credit card companies to cardholders for a particular month):

- BillAmt1 - BillAmt6: Amount of bill statement (NT dollar). BillAmt1 = amount of bill statement in September 2005; BillAmt2 = amount of bill statement in August 2005; . . .; BillAmt6 = amount of bill statement in April 2005.

The last set of variables takes into account the amount of previous payments made in a specific month.

- PayAmt1 - PayAmt6: Amount of previous payment (NT dollar). PayAmt1 = amount paid in September, 2005; PayAmt2 = amount paid in August, 2005; . . .; PayAmt6 = amount paid in April 2005.

The dataset originates from a study aimed at predicting credit card defaults, by [**?**], where the data came from an important bank in Taiwan. The study was aimed at comparing various classification techniques to predict default payments by credit card clients. Given it is sensitive information the data is completely anonymized and aggregated, which means that individual identities are not revealed and the data is presented in a summary form e.g. clients are identified in an anonymized chronological order from 1 to 30 000. Each observation is ordered according to the features described above.

2.2 Response Variable Distribution

Among the 30,000 observations, 6636 observations (22.12%) are cardholders with default payment which means that 77.88% of observations are clients who did not default, this is shown in Figure 1. This discrepancy might stem from significant bias, thus not accurately reflecting the bank's clientele. Nonetheless, it's important to recognize that the data collection occurred during a period of debt crisis, reinforcing the view that the data constitutes an unbiased sample of the customer base[8]. Therefore, it is crucial to consider the high rate of defaults when drawing conclusions from the research presented. If the class imbalance issue is not properly handled, classification algorithms will tend to prioritize the majority class while neglecting the minority class, leading to suboptimal performance in accurately identifying instances of the underrepresented class[2]. To deal with the class imbalance we employed the oversampling technique Synthetic-Minority Oversampling Technique, for which the details can be found in the appendix over here.

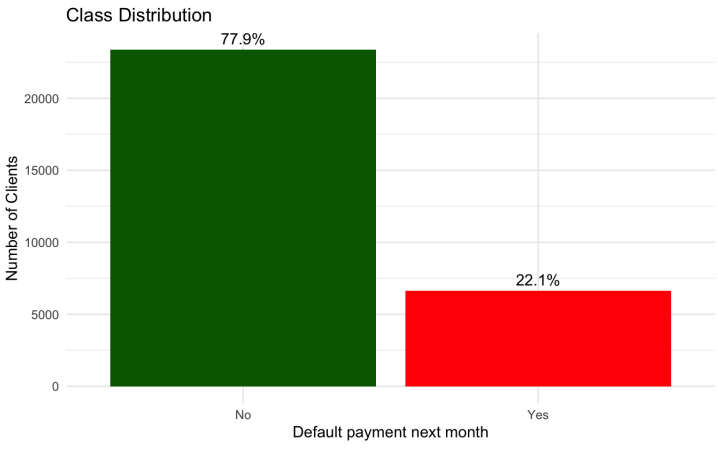

Fig. 1: Response Variable Distribution

2.3 Summary Statistics

As we are concerned with credit card default risk segmentation, we performed some summary data analysis on the credit limit balance and credit utilization patterns.

Credit Limit Overwiew

We performed a 5-number summary shown in Figure 2, and we observed the following.

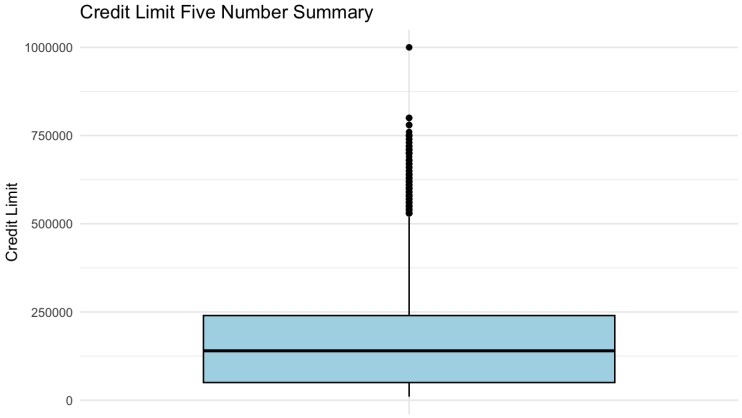

Fig. 2: Credit Limit: Five Number Summary

- The minimum credit limit balance was NT$ 10 000
- 25% of customers had credit limit balances of NT$ 50 000 or less, indicating that 75% of clients had credit limit balances above NT$ 50,000.
- The median credit limit was NT$ 140 000, indicating that 50% of customers have credit limit balances less than NT$ 140 000 whilst another 50% of customers have credit limit balances above NT$ 140 000.
- 75% of customers had credit limit balances below NT$ 240 000, whilst 25% of customers had credit limit balances above NT$ 240 000
- The maximum credit limit amount was NT$ 1 000 000
- The average male credit limit balance was NT$ 163519.82 and the average female credit limit balance was NT$ 170086.46
- Customer ages ranged from 21 to 79 years old, with the average age being 35 years old.

The summary above suggests the following; there is significant variability in the credit limit balances that are given to customers as the limits range from NT$ 10 000 to NT$ 1 000 000. Furthermore, we note that the median value is closer to the third quantile than it is to the first, indicating that the data is right-skewed, which means that there is a large number of customers who have credit limits on the lower end of the range, with fewer individuals having higher credit limits. The significant difference between the maximum value and the third quantile finally indicates that outliers exist on the higher end of the credit limit balances.

Credit Utilization Patterns

Analysing overall credit utilization shown in Figure 3, we preliminary noted the following:

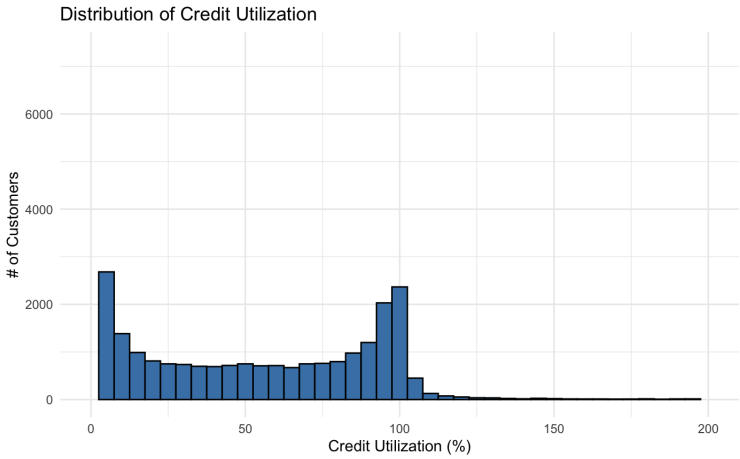

Fig. 3: Credit Utilization

- There are a few customers that have exceeded their credit limit facility. Indicating several default risk customers.
- Overall the distribution is left-skewed, indicating that more customers have utilized a large portion of their credit facility than less. This could indicate that several customers could be at risk of defaulting as they owe more money, this will prove interesting when we build the credit default risk segmentation model.

2.4 Correlation Analysis

A correlation analysis was performed to assess whether various pairs move in coordination with one another. Positive correlations indicate that the features are moving in the same direction and negative correlations indicate that features are moving in the opposite direction. In Figure 4 below we show a correlation matrix where the correlations were calculated using the Pearson Correlation (p)

$$p(X,Y) = \frac{Cov(X,Y)}{\sigma_x \sigma_y} \quad where -1 \leq p \leq 1 \qquad (1)$$

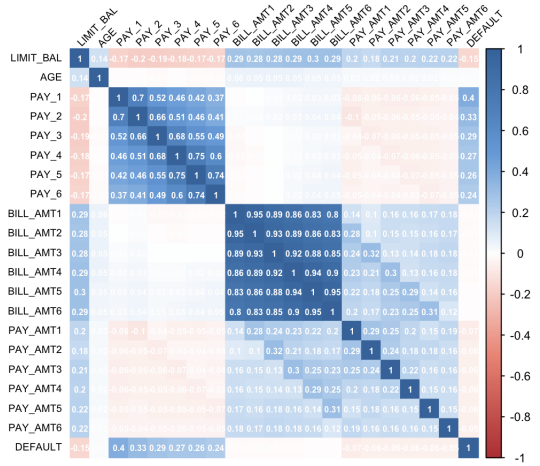

Fig. 4: Correlation Analysis

Observing the correlations in Figure 4, we note the following:

- There is a noticeable correlation between the PAY_n features
- The BILL_AMT_n features have really high correlation, noticeably BILL_AMT_1 and BILL_AMT_2 have a correlation of 0.95 BILL_AMT_2 and BILL_AMT_3 have a correlation of 0.93 and finally BILL_AMT_3 and BILL_AMT_4 have a correlation of 0.92
- Correlation with the target variable (DEFAULT) is low across the features, indicating that no one feature disproportionately affects the target variable, therefore indicating that there is no strong data redundancy.

The correlation plot above shows that correlation exists between the features, in order to remove correlated features we will be performing various dimension reduction techniques as we build our credit default risk segmentation model.

3 Customer Segmentation

Customer segmentation aims to establish a framework for distinguishing among clusters of customers rather than individuals [8]. Within the scope of this study, it enables the forecasting and evaluation of the financial consequences of initiatives that impact broad segments of customers at once, thereby fostering a more comprehensive strategy for decision-making. Credit card segmentation is not a new problem and previous studies have implemented methods such as Self-Organizaing Maps and K-Means Clustering[5][12]. We will be employing k-means to apply customer segmentation as well, however, our implementation will be based on various dimension reduction techniques, as we wish to assess the customer segments provided by each dimension reduction technique.

3.1 K-Means Clustering

The core concept underlying k-means clustering is the establishment of a mean or average value denoted by K, around which the data can be grouped. We do this by reassigning every point from the initial, suboptimal clustering to the cluster centre that is closest to it. Subsequently, revise the cluster centres by calculating the mean of the points comprising them. Maintain this iterative process of centre updating and reassignment until the convergence criteria are satisfied [**?** **?**]. Obtaining the optimal k-clusters by minimizing the Within-Cluster-Sum-of-Squares (WCSS), defined below, is how we aim to construct the various customer segment clusters.

$$\text{WCSS} = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2 \tag{2}$$

where:

- k is the number of clusters,
- C_i is the set of points that belong to cluster i,
- x is a point in cluster C_i,
- μ_i is the centroid of cluster C_i,
- $||x - \mu_i||^2$ is the squared Euclidean distance between a point x and the centroid μ_i of its cluster.

3.2 Building Credit Default Risk Segments

Several customer segments can be built, however, as we are concerned with credit card default risk management, we built a novel credit default risk customer segmentation model. Based on literature factors such as bill-payment history and one's credit utilization ratio are the most highly weighted factors when assessing one's credit default risk[6]. To this end, our focus will be to group customers in each cluster into credit default risk segments based on similar factors, in our case this will include the following:

- Average Credit Utilization
- Average Number of Late Payments

As a result of the above and using thresholds inspired by general credit default risk literature[6] we defined and built the following credit default risk segments:

- **Low Risk:** For customers with no late payments in the past 6 months and a low credit utilization threshold of no more than 30%.
- **Medium Risk:** One or two late payments in the past 6 months and a moderate credit utilization threshold of over 40% but less than 75%.
- **High Risk:** More than two late payments in the past 6 months and high credit utilization threshold of 75% or more.

We start our journey building the various customer default risk segments with the combination of PCA and K-Means, followed by Autoencoders and K-Means and finally Variational Autoencoders and K-Means.

3.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is an unsupervised learning algorithm that reduces the dimensionality of the data while preserving as much 'variability' (i.e., statistical variance) as possible. It identifies the directions in which the data varies the most, denoted as principal components (PCs). These components are linear combinations of the original variables and are orthogonal to each other, each representing a different proportion of the total variance.

Formally, let X be a data matrix with n observations and p variables, centered by subtracting the mean of each variable. The covariance matrix C of X is given by:

$$C = \frac{1}{n-1} X^T X$$

PCA seeks to find the eigenvalues λ_i and corresponding eigenvectors v_i of C, satisfying:

$$Cv_i = \lambda_i v_i$$

The eigenvectors are called the principal axes or principal directions of the data. The eigenvalues represent the amount of variance captured by each axis. Typically, the eigenvectors are ordered by decreasing eigenvalues to prioritize the directions with the most significant variance.

The principal components Z are then computed as:

$$Z = XV$$

where V is the matrix of selected eigenvectors (principal axes).

The first principal component Z_1 is a line that best represents the data in the least-squares sense, and is calculated as:

$$Z_1 = Xv_1$$

where v_1 is the eigenvector associated with the largest eigenvalue λ_1. Each subsequent component Z_i is calculated in the same way, subject to the constraint that it is orthogonal to the preceding components.

In summary, PCA transforms the original correlated variables into a set of uncorrelated principal components ordered by the variance they explain, starting from the greatest to the least[10].

As eluded earlier our usage of it will be to determine the optimal number of k-means clusters using the optimal number of Principal Components(PCs), i.e. the PCs the explain the most variability. To do this we review the PCA Scree Plot in Figure 5 which consists of 26 principal components.

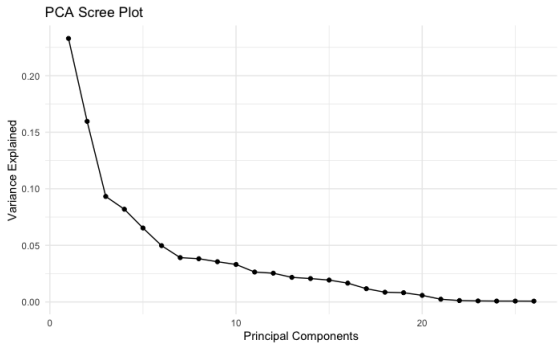

Fig. 5: PCA Scree Plot

- We note that the variance explained by each principal component decreases almost monotonically, as expected.
- There seems to be an "elbow" after the third component, suggesting that most of the useful information is contained in the first 3 components, however, these components only capture 48.57% of the variance, suggesting that the "elbow" is not indicative of the most significant principal components.
- There is another "elbow" at the 7th principle component however the variability here is about 72.19%. Both of these observed "elbows" are not very pronounced, indicating that the technique is not entirely capturing the structure of the data and as a result we note that the variance is somewhat evenly spread across the components. To capture about 90% of the variance we need the first 13 components, i.e. half the components.
- This evenly spread variance may indicate that the linear projection is not sufficient to capture the data pattern.

PCA Credit Default Risk Customer Segments

Given we require 13 PCs to capture about 90% the variability within the data, the k-means algorithm was executed with these 13 PCs as the input data. Figure 6 illustrates a WCSS Elbow Method plot to determine the optimal number of clusters to group customers.

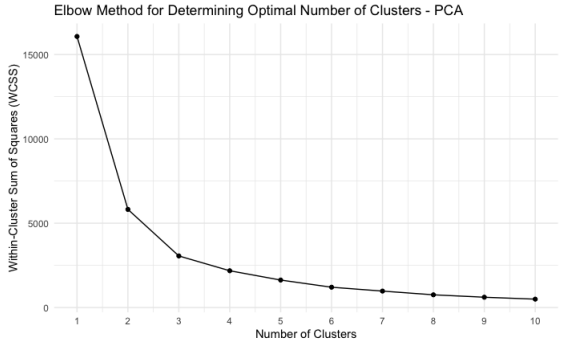

Fig. 6: PCA K-Means Elbow Method

We observe that the optimal number of clusters is roughly 3, as after this point the decrease in WCSS is not that significant.

Using these three clusters we analysed how well these clusters are separated in order to determine if the relationships within the data are being well accounted for. This was observed in a three-dimensional latent space, using the first 3 principle components, which have a variance contribution of 48.57% and an MSE of 4.24.

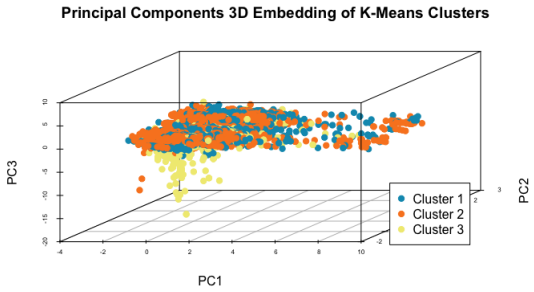

Fig. 7: PCA 3D K-Means Clusters

Reviewing Figure 7 we note that we were able to find distinct clusters, however, these clusters are not exactly well separated as there appears to be some overlap, especially between Cluster 1 and Cluster 2. Cluster 3 seems

to have a visible distinction but not very well. Finally, we note that due to the sub-optimal cluster separation, PCA might not be capturing the underlying relationship in our non-linear hypothesised data set.

Let's now analyse how the customers are grouped within these clusters under the proposed credit default risk segmentation model.

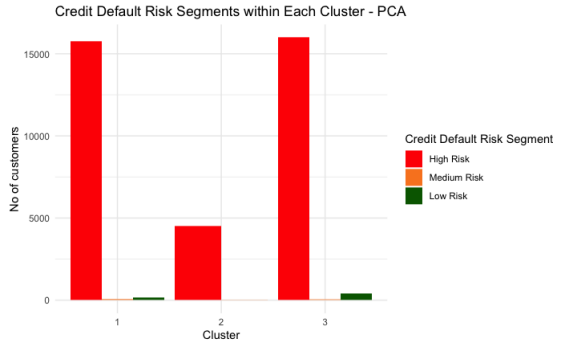

Fig. 8: PCA Customer Segmentation

Reviewing Figure 8, we observed that Cluster 1 and Cluster 3 are very similar, cluster 2 hardly has any non-high-risk clients. To this end, we note that two credit default risk segments namely "Risky" and "Non-Risky" would suffice to segment the pool of customers within the said data set. This again enforces the idea that there was not much separation within the clusters resulting from k-Means using PCA.

We now turn our attention to building customer segments using neural networks, starting first with Autoencoders and then Variational Autoencoders.

3.2.2 Autoencoders

Autoencoders are a family of neural networks for which the input is the same as the output. They compress the input into a lower-dimensional code and then reconstruct the output from this representation. The code is a compact "summary" or "compression" of the input, also known as the latent-space representation.

An autoencoder has two main parts: the encoder and the decoder. The encoder compresses the input and produces the code, the decoder then reconstructs the input only using this code. Formally, given an input $x \in \mathbb{R}^d$, the encoder $f : \mathbb{R}^d \to \mathbb{R}^p$ maps the input to the code, and the decoder $g : \mathbb{R}^p \to \mathbb{R}^d$ maps the code to the reconstruction $\hat{x}$, where typically $p < d$.

The autoencoder learns to minimize the difference between x and its reconstruction $\hat{x}$, typically measured by a loss function such as the mean squared error:

$$L(x, \hat{x}) = ||x - \hat{x}||^2$$

Autoencoders are trained through backpropagation, adjusting the weights of both the encoder and decoder to minimize the reconstruction loss.

Below is a diagram of a basic autoencoder:

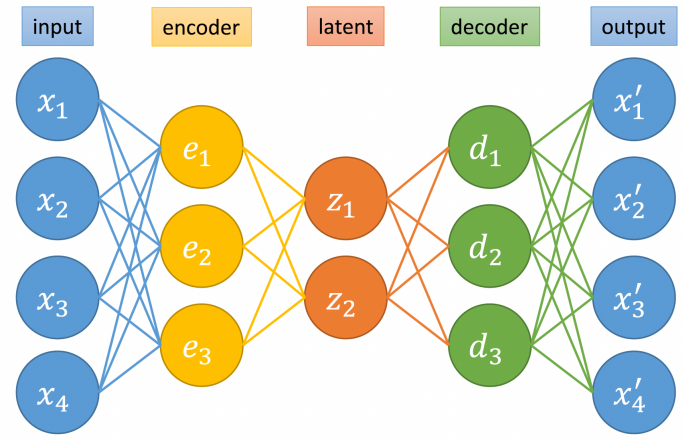

Fig. 9: Schematic of an Autoencoder

The potential of autoencoders goes beyond just dimensionality reduction; they can be used for denoising data, feature extraction, and even generative models in the form of Variational Autoencoders (VAEs)[1], which we will explore later.

We tested various autoencoder architectures during our hyperparameter tuning evaluation, for brevity we provide information for the architecture that performed best with regards to minimising the reconstruction loss.

Layer	Neurons	Activation	Description
Input	20	-	Input Features
Encoder	13	Tanh	Single hidden layer
Latent	1 to 12	-	Compressed representation
Decoder	13	Tanh	Reconstruction from latent space
Output	20	-	Reconstructed output

Tab. 1: Autoencoder architecture with varying latent layer dimensions.

The autoencoder performed best when we had 12 neurons in the latent layer i.e. code size of 12, the achieved MSE at this code size was 0.1483 which is lower than the 2.557 MSE achieved with 13 PCs (caputered 90% variance). Indicating that the non-linear method was able to better account for the underlying relationships within the data.

Autoencoder Credit Default Risk Customer Segments

Similar to PCA, we analyzed what patterns we could observe when performing k-means clustering with the latent encoded representation values at the code size of 12 (minimum MSE). Starting our review with the optimal number of clusters, Figure 10 shows using two clusters would be optimal.

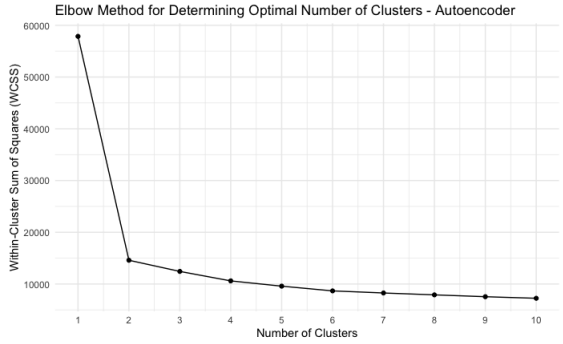

Fig. 10: AE K-Means Elbow Method

Using these two clusters we analyzed their representation using the first three-dimension of the 12-dimension latent space compression, this is shown in Figure 11 below.

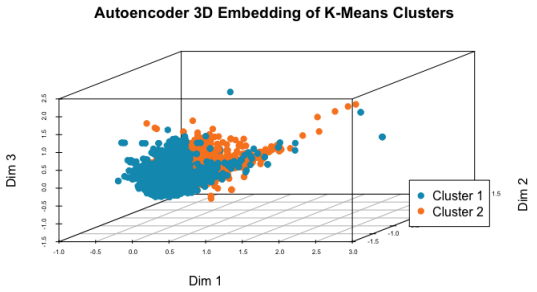

Fig. 11: AE 3D K-Means Clusters

Compared to PCA - we observed the presence of well-defined clusters. With an MSE of 0.1483 compared to PCA's MSE of 2.5573 with 13 PCs, we note that the autoencoders have done a decent job of grouping customers into various clusters.

However, we note some disjointed points within cluster 2, this is a result of one of the short-falls of autoencoders i.e the latent space is not regularized, meaning close points do not always get mapped together, for customer segmentation to be optimal we need closely related points to be mapped together. Let's analyze how these two clusters group customers under the proposed credit default risk segmentation, this is shown in Figure 12.

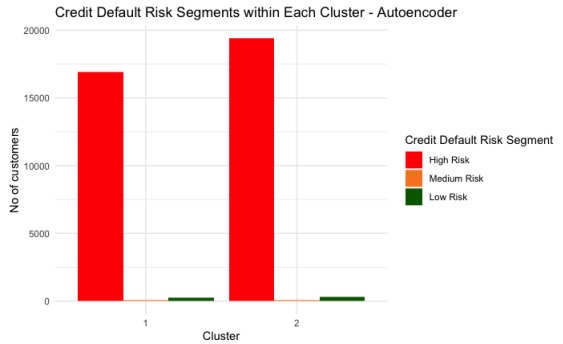

Fig. 12: AE Customer Segmentation

We still observe very low levels of low and medium-risk clients within the clusters. The high-risk customers seem to be evenly spread across the two clusters similar to the the PCA segmentation. This further suggests that two segments "Risky" and "Non-Risky" would have been sufficient. Finally, due to the observed similarity between the two clusters, it doesn't seem like the clusters are providing significantly different information and could potentially be combined into one, which would defeat the segmentation.

3.2.3 Variational Autoencoders

Variational Autoencoders are the final dimension reduction technique that will be examined in the context of customer segmentation and prediction. Variational Autoencoders (VAEs) embody a robust amalgamation of neural networks and variational inference, presenting substantial progressions in comparison to conventional autoencoders. VAEs alter this paradigm from the traditional goal of learning a deterministic mapping from the input space to a compressed latent space and backwards (autoencoders). The encoding process is understood as a model that acquires knowledge of the parameters of probability distributions, which are commonly Gaussian in nature, and which represent the latent space data.

The innovation introduced by VAEs lies in their treatment of the latent space. Instead of encoding an input x to a single point, a VAE encodes it to a distribution described by two vectors: means μ and standard deviations σ, which define a Gaussian distribution in the latent space. The actual encoding z for a given input is then sampled from this distribution:

$$z \sim \mathcal{N}(\mu(x), \sigma(x)^2)$$

The decoder part of the VAE uses this sampled z to reconstruct the input data. This introduces regularization via the Kullback-Leibler (KL) divergence, encouraging the learned distributions to approximate the prior distribution, typically a standard Gaussian. This divergence acts as a penalty for deviating from the prior, effectively preventing overfitting and ensuring a well-structured latent space.

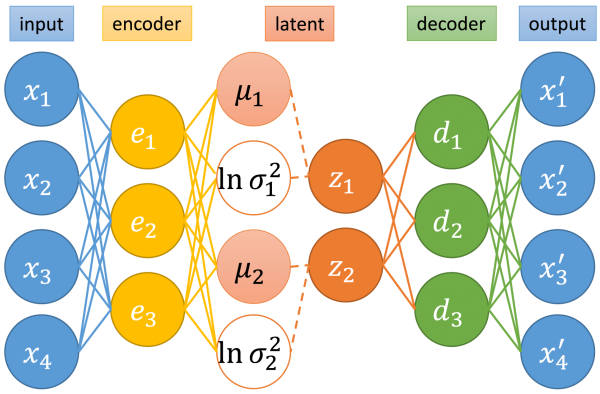

Fig. 13: Schematic representation of a Variational Autoencoder. The encoder maps input data to a distribution in latent space, from which the decoder reconstructs the input.

Formally, the VAE is trained by maximizing the ev-

idence lower bound (ELBO) on the marginal likelihood of the observed data, which involves two terms: the reconstruction loss and the KL divergence, balancing data fidelity with regularization:

$$\text{ELBO}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \text{KL}(q_\phi(z|x) \,||\, p(z))$$

where $q_\phi(z|x)$ is the encoder's approximate posterior distribution, $p_\theta(x|z)$ is the decoder's likelihood, $p(z)$ is the prior over the latent variables, and θ, ϕ are the parameters of the decoder and encoder, respectively.

By optimizing the ELBO, VAEs effectively solve several challenges associated with traditional autoencoders. They prevent overfitting by penalizing encodings that stray too far from the prior distribution, thus fostering a more regularized latent space. This regularization also ameliorates the problem of "over-confident" representations, where a standard autoencoder might map vastly different inputs to the same encoding, losing information in the process. Moreover, VAEs facilitate a smooth and continuous latent space that ensures small changes in the latent variables lead to small changes in the reconstructed inputs[1].

The variational autoencoder we constructed was as per the architecture described in Table 2 below:

Layer	Neurons	Activation	Description
Input	20	-	Input features.
Encoder	13	tanh	Encodes to compressed representation.
μ	1-12	-	Mean vector of the latent space distribution, used for defining the center of the latent space Gaussian distribution.
σ	1-12	-	Standard deviation (log variance) vector of the latent space distribution, used for defining the spread or dispersion of the latent space Gaussian distribution.
Sampling	1-12	-	Samples a latent variable from the Gaussian distribution defined by μ and σ. This sampling introduces the stochasticity that is characteristic of VAEs, enabling the generation of diverse outputs.
Decoder	13	tanh	Decodes the sampled latent variables from the latent space back into a reconstruction of the original input data.
Output	20	-	The final layer that outputs the reconstructed version of the input data, attempting to match the original input as closely as possible.

Tab. 2: Variational Autoencoder Architecture

Similar to the autoencoder the variational autoencoder performed best when we had 12 neurons in the latent layers i.e. code size of 12, the achieved MSE at this code size was 0.403 which is lower than the 2.557 MSE achieved with 13 PCs (captured 90% variance) but higher than the 0.148 achieved by the autoencoders. We have to take into account that variational autoencoders have a regularization penalty, so even though the MSE was higher than autoencoders, it doesn't necessarily mean it's inferior, especially given that the MSE difference is not proportionally different. Nonetheless, we note that as a non-linear method, it outperformed the linear PCA method.

Variational Autoencoders Credit Default Risk Customer Segments

Analysing our final customer segment strategy using variational autoencoders, we aimed to seek the optimal number of clusters for this setup which proved tricky. However, we opted for 3 clusters as the WCSS reduction after this stage decreases equally, this is shown in Figure 14.

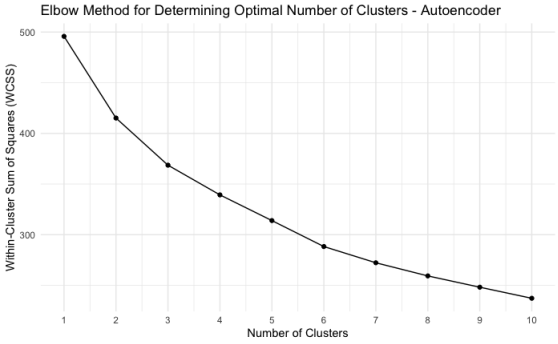

Fig. 14: VAE K-Means Elbow Method

Mapping these clusters to the 12-dimensional (minimum MSE of 0.403) latent space representation and plotting its first three dimensions. We observed the following, as shown in Figure 15.

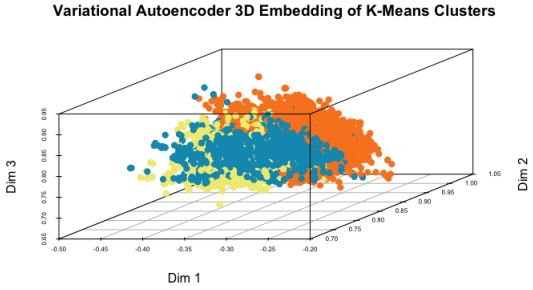

Fig. 15: VAE 3D K-Means Clusters

Reviewing the plot, we note that not only can we observe distinct clusters, but any "holes" have been accounted for. This indicates that the variational autoencoders have maintained any underlying "close" relationships within the data whilst still being able to produce well-separated clusters, which is a bonus for customer segmentation. Let's finally analyze how it performs under our credit default risk segment model.

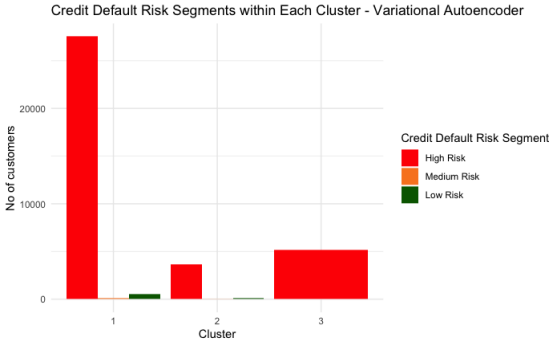

Fig. 16: VAE Customer Segmentation

What do we notice? Well, the first cluster has almost over 80% of customers under high risk, which means that this individual cluster could be termed the high-risk customer segment cluster and the rest could be aggregated into a separate non-high-risk cluster. The distinction is important here, previously it was not easy to separate the

clusters under our credit default risk model, as the high-risk segment was mostly spread out between 2 clusters for both the PCA and AE segmentation models. With the variational autoencoder strategy, we note that the actual clusters have an underlying mapping with the credit default risks, so we could say cluster 1 maps one-to-one with high risk and aggregate clusters 2 and 3 such that they are non-high risk, this creates a non-complicated credit default risk segmentation model.

4 Predicting Credit Card Defaults

Following the establishment of the elementary credit default risk segmentation model utilizing dimensionality reduction techniques such as Principal Component Analysis (PCA), Autoencoders (AE), and Variational Autoencoders (VAE), our next research phase focuses on answering the second question: Can we create a dependable model for predicting defaults based on the best customer segmentation model we found? This section explores the use of Support Vector Machines (SVMs) for predicting credit card defaults. It utilizes the modified feature spaces obtained from the dimensionality reduction approaches outlined before.

4.1 Support Vector Machines

Support Vector Machines (SVMs) are a robust category of supervised learning models employed for problems including classification and regression. They are especially suitable for binary classification tasks. SVMs fundamentally seek to identify the most advantageous hyperplane that effectively divides the data points belonging to distinct classes, while maximizing the separation distance. The ideal hyperplane is defined as the one that maximizes the distance between itself and the nearest data points of any class, which are called support vectors.

The SVM model in a linearly separable situation is defined by the optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 \quad \text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1, \forall i \qquad (3)$$

where $\mathbf{w}$ represents the weights, b the bias term, $\mathbf{x}_i$ the input features, and y_i the class labels.

However, many real-world datasets are not linearly separable. SVMs address this challenge through the use of kernel functions, enabling the model to operate in a transformed feature space where a linear separation is possible. The kernel trick involves mapping input features into high-dimensional spaces without explicitly performing the transformation. Common kernel functions include:

- Linear Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
- Polynomial Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)^d$, where d is the degree of the polynomial, γ is the scale factor, and r is the offset.
- Radial Basis Function (RBF) Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)$, where γ is the scale parameter.
- Sigmoid Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)$

SVMs may effectively carry out non-linear classification by utilizing kernel functions, enabling them to capture intricate correlations within the data. The combination of this attribute, along with the model's ability to generalize well, renders SVMs a resilient tool for a diverse range of machine learning applications [7].

4.2 Predictive Model

For our predictive model on credit card default prediction, we chose to use the Radial Basis Function (RBF) kernel for our predictive model. The RBF kernel, denoted as $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)$, is particularly effective in capturing non-linear correlations by transforming input features into a higher-dimensional space. The selection of the model is based on the sophisticated data patterns that influence credit default risk, requiring a model that can comprehend these complex elements beyond simple linear separability. The RBF kernel, with its two crucial parameters C and γ, provides a trade-off between model flexibility and the prevention of overfitting, making it an attractive choice for our investigation.

4.3 Customer Segmentation vs No-Customer Segmentation Prediction Performance

We present the results of our default predictive model. The model was trained and tested under a no-customer segmentation and a customer segmentation framework. With the main goal to evaluate how effective the proposed credit default risk segmentation performs when predicting if a customer will default or not. No-customer segmentation essentially reflects the original setup i.e. no clusters (simply the dimension reduction), or one single cluster, whereby the customer-segmentation framework is as discussed earlier.

For our model parameters, inspired by research[9] where SVMs have been implemented to predict on credit default we set the parameters $C = 100$ and $\gamma = 0.1$ as shown in Table 3. We measure the performance of our models based on their Accuracy, Recall, Precision, F1-Score and AUC. For each model we used the optimal compression for each technique in order to train and test, that is for PCA we used the first 13 PCs, AE and VAE we used the compression at code size 12. Before proceeding to the results below we give a brief description of each performance evaluation metric.

- **Accuracy**: The proportion of true results (both true positives and true negatives) in the dataset.

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

  High accuracy means that the model correctly predicted most of the positive and negative cases.

- **Recall (Sensitivity)**: The proportion of actual positives correctly identified by the model.

$$\text{Recall} = \frac{TP}{TP + FN}$$

High recall indicates that the model is good at capturing the majority of the positive cases.

- **Precision**: The proportion of positive identifications that were actually correct.

$$\text{Precision} = \frac{TP}{TP + FP}$$

High precision means that when the model predicts a positive result, it is likely to be correct.

- **F1 Score**: The harmonic mean of Precision and Recall.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

A high F1 score suggests a balanced model with good precision and recall, useful when classes are imbalanced.

- **AUC (Area Under the Curve)**: The area under the Receiver Operating Characteristic (ROC) curve.

$$\text{AUC} = \int_{x=0}^{1} \text{ROC}(x)\, dx$$

A high AUC value indicates that the model has a good measure of separability, with a high ability to distinguish between positive and negative classes.

No-Customer Segmentation Prediction Performance

Initial analysis of the no-customer segmentation results shows that SVM+VAE garnered the best result with an overall F1-score of 0.8709, however, it was not significantly better than its SVM+AE counterpart.

Technique	Parameters	F1-score
PCA	kernel: RBF, C: 100, γ: 0.1	0.7981
AE	kernel: RBF, C: 100, γ: 0.1	0.8678
VAE	kernel: RBF, C: 100, γ: 0.1	0.8709

Tab. 3: SVM F1-Scores with Dimension Reduction Technique

Examining the other metrics under the no-customer segmentation framework as shown in Table 4. We note that SVM+PCA had the poorest accuracy whereas SVM+AE and SVM+VAE had the same performance. Although the models SVM+AE and SVM+VAE have similar accuracy levels we would recommend SVM+VAE as a credit default prediction model as it marginally outperforms SVM+AE on the other core metric, that being the F1-score.

Technique	Accuracy	Recall	Precision	F1-score	AUC
PCA	0.6771667	0.8247201	0.7732687	0.7981661	0.4983000
AE	0.7940000	0.8845823	0.8544093	0.869234	0.6842000
VAE	0.7903333	0.914298	0.8315707	0.8709744	0.6400000

Tab. 4: SVM Performance with no-customer segmentation

Customer Segmentation Prediction Performance

We now proceed to analyse the prediction ability of our model under a customer segmentation framework, that is assessing its ability to predict credit default per cluster as opposed to the opposite. We summarise our findings in Table 5 and comment as follows:

- **PCA+K-Means:** Following our analysis of the clusters formed by PCA+K-means we note the poor cluster separation has led to poor model performance results, making it difficult to suggest this method, because for example in cluster 2, the bank would only have a credit default prediction accuracy of 25.92%, which is extremely lacking.
- **AE+K-Means:** For this setup we recall that we noted that the two clusters were very similar and could be aggregated into one, this is further enforced as we observe very similar performance across the model metrics. Notably, we observe that the individual cluster accuracy is lower than their accuracy when combined besides all other metrics being similar.
- **VAE+K-Means**: In this final customer segmentation setup, we observe that one cluster (Cluster 1) had really good performance metrics whereas Cluster 2 and Cluster 3 had sub-optimal results. This was expected as we noted that Cluster 1 had over 80% of the customers and therefore had an observed one-to-one mapping to the High-Risk credit default risk segment. As a result with fewer customers, clusters 2 and 3 were bound to perform poorly. With this in mind, we observed that cluster 1 outperformed the single cluster SVM+VAE model across all metrics and notably had an accuracy of 82.38%. Given that cluster 1 had the majority of customers, we noted earlier that we could simply combine clusters 2 and 3, this was done and it greatly improved our prediction model as their combined result yielded superior results to their individual counterparts. This was powerful as we note that with this combination, simply having cluster 1 as a segment (e.g. "Risky") and clusters 2+3 as another segment (e.g. "Non-Risky") i.e. **two clusters** in the total, that this setup outperforms any other setup, be it the AE+K-Means clusters or the no-customer segment model version of SVM+VAE.

Segmentation	Cluster	Accuracy	Recall	Precision	F1-score	AUC
PCA + K-Means	Cluster 1	0.6724138	0.7906977	0.7727273	0.7816092	0.5620155
	Cluster 2	0.2593760	0.1194636	0.6163522	0.2001361	0.4312206
	Cluster 3	0.7116894	0.8905420	0.7730627	0.8276543	0.4888702
AE + K-Means	Cluster 1	0.7671697	0.9553116	0.7914907	0.8657194	0.5164489
	Cluster 2	0.7598983	0.9976111	0.7609329	0.8633450	0.5018358
VAE + K-Means	Cluster 1	0.8238994	0.8909884	0.9041298	0.8975110	0.6417559
	Cluster 2	0.4651163	0.3166227	0.3738318	0.3428571	0.4493716
	Cluster 3	0.4724971	0.4414314	0.8430326	0.5794495	0.5293094
	Cluster 2+3	**0.8289883**	**0.9955451**	**0.8315707**	**0.9062000**	**0.5063000**

Tab. 5: SVM Performance with Customer Segmentation

5 Conclusion

The combination of Variational Autoencoders (VAE) and K-Means clustering has shown potential for customer seg-

mentation in an elementary credit default risk model. Our analysis suggests that banks can use the strategy of stratifying consumers into 'risky' and 'non-risky' segments based on the dataset size. By doing so, they can determine the probability of default with an accuracy of 82.38% specifically for the 'risky' segment and an accuracy of 82.89% for the 'non-risky' segment.

The results indicate that the VAE+K-Means segmentation technique has the potential to improve credit risk management. These findings, obtained from a very small dataset, imply that additional exploration of the VAE+K-Means strategy with bigger datasets is warranted. Although the current model offers foundational understanding, its positive results support the need for additional investigation and improvement in order to achieve the highest level of predicted accuracy.

6 Bibliography

[1] Dimensionality reduction of sdss spectra with variational autoencoders, 2020.

[2] T. M. Alam, K. Shaukat, I. A. Hameed, S. Luo, M. U. Sarwar, S. Shabbir, J. Li, and M. Khushi. An investigation of credit card default prediction in the imbalanced datasets. *IEEE Access*, 8:201173–201198, 2020.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002.

[4] B. Emil Richard Singh and E. Sivasankar. Risk analysis in electronic payments and settlement system using dimensionality reduction techniques. In *2018 8th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pages 14–19, 2018.

[5] C.-C. W. Hui Wu. Customer segmentation of credit card default by self organizing map. 2018.

[6] J. Kagan. Default risk: Definition, types, and ways to measure, 2023.

[7] A. Mammone, M. Turchi, and N. Cristianini. Support vector machines. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):283–289, 2009.

[8] M. Merikoski, A. Viitala, and N. Shafik. Predicting and preventing credit card default. 2018.

[9] M. Merlo. Default of credit card clients, 2022.

[10] S. Shalev-Shwartz and S. Ben-David. *Dimensionality Reduction*, page 278–294. Cambridge University Press, 2014.

[11] I.-C. Yeh and C. hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2, Part 1):2473–2480, 2009.

[12] R. Z. Ying Chen. Research on credit card default prediction based on k-means smote and bp neural network. 2021.

7 Appendix

7.1 Data Cleaning

Upon performing a high-level data review, we observe from Figure 17 below that a few attributes have discrepancies. These are highlighted below.

Features Summary Statistics

	vars	n	mean	sd	min	max	range	se
ID	1	30000	15000.50	8660.40	1	30000	29999	50.00
LIMIT_BAL	2	30000	167484.32	129747.66	10000	1000000	990000	749.10
SEX	3	30000	1.60	0.49	1	2	1	0.00
EDUCATION	4	30000	1.85	0.79	0	6	6	0.00
MARRIAGE	5	30000	1.55	0.52	0	3	3	0.00
AGE	6	30000	35.49	9.22	21	79	58	0.05
PAY_0	7	30000	-0.02	1.12	-2	8	10	0.01
PAY_2	8	30000	-0.13	1.20	-2	8	10	0.01
PAY_3	9	30000	-0.17	1.20	-2	8	10	0.01
PAY_4	10	30000	-0.22	1.17	-2	8	10	0.01
PAY_5	11	30000	-0.27	1.13	-2	8	10	0.01
PAY_6	12	30000	-0.29	1.15	-2	8	10	0.01
BILL_AMT1	13	30000	51223.33	73635.86	-165580	964511	1130091	425.14
BILL_AMT2	14	30000	49179.08	71173.77	-69777	983931	1053708	410.92
BILL_AMT3	15	30000	47013.15	69349.39	-157264	1664089	1821353	400.39
BILL_AMT4	16	30000	43262.95	64332.86	-170000	891586	1061586	371.43
BILL_AMT5	17	30000	40311.40	60797.16	-81334	927171	1008505	351.01
BILL_AMT6	18	30000	38871.76	59554.11	-339603	961664	1301267	343.84
PAY_AMT1	19	30000	5663.58	16563.28	0	873552	873552	95.63
PAY_AMT2	20	30000	5921.16	23040.87	0	1684259	1684259	133.03
PAY_AMT3	21	30000	5225.68	17606.96	0	896040	896040	101.65
PAY_AMT4	22	30000	4826.08	15666.16	0	621000	621000	90.45
PAY_AMT5	23	30000	4799.39	15278.31	0	426529	426529	88.21
PAY_AMT6	24	30000	5215.50	17777.47	0	528666	528666	102.64
default payment next month	25	30000	0.22	0.42	0	1	1	0.00

Fig. 17: Feature Summary Statistics

The education, marriage and PAY_n attributes have unreported categories.

- The education attribute has 3 unreported categories, as we note that in the dataset it can take the values 0,1,2,3,4,5,6. However, the only reported categories were 1,2,3,4.
- The marriage attribute we note that it can additionally take the value 0, whereby it was reported to take the values 1,2 and 3.
- The PAY_n attribute is expected to only present the values -1,1,2,3,4,5,6,7,8,9, however in the dataset it additionally takes the values -2 and 0.

We proceeded to clean this as follows:

1. For the education attribute, the unreported 0,5,6 categories were added to the "other" category reported by 4.
2. For the marriage attribute the unreported 0 category was added to the "other" category reported by 3.
3. For the PAY_n attribute all the values -2, -1 are mapped to 0, such that the variable is not categorical and neatly indicates how many months a payment was delayed.

7.2 Data Preprocessing

7.2.1 One-Hot Encoding

The categorical variables EDUCATION, SEX and MARRIAGE have been encoded with integer numbers which means they can be easily fed to a machine learning algorithm. Nonetheless, these are nominal features meaning that we can't treat them with an implied order as this could lead to suboptimal results. Utilizing one-hot encoding eliminates any implied order among these categories, ensuring no ordinal relationships are assumed where none exist.

This method involves generating a new dummy variable for every distinct value within a nominal feature column. Binary values are subsequently employed to signify the specific class to which an example belongs. After performing this technique we dropped the EDUCATION, SEX and MARRIAGE features and introduced the following:

Feature	Value
MALE:	1 = male; 0 = female.
MARRIED:	1 = married marital status; 0 = otherwise.
GRAD_SCHOOL:	1 = graduate school level of education; 0 = otherwise.
UNIVERSITY:	1 = university level of education; 0 = otherwise.
HIGH_SCHOOL	1 = high school level of education; 0 = otherwise.

Tab. 6: One-Hot Encoded Values

Performing a one-hot encoding does not cause information loss as the data is translated from the originally observed data format in Table 6 to a new format as shown in Table 7.

ID	SEX	MARRIAGE	EDUCATION
0	1	1	1
1	2	2	2
2	1	3	3
3	2	1	4

Tab. 7: Data Format before one-hot encoding

ID	MALE	MARRIED	SINGLE	GRAD SCHOOL	UNIVERSITY	HIGH SCHOOL
0	1	1	0	1	0	0
1	0	0	1	0	1	0
2	1	0	0	0	0	1
3	0	1	0	0	0	0

Tab. 8: Data Format after one-hot encoding

7.2.2 Data Partition

To execute the proposed machine learning models, we need to partition our data set into training and test data sets. The training set was utilised to train the models, whereas the test set was used to assess model performance.

The data was split into 80% training and 20% testing. The split maintained the imbalanced class distribution for consistency. Table 9 shows a summary of the training and test data split to this effect.

	Non-defaulter	Defaulters	Total
Training set	18720(78%)	5280(22%)	24000
Test set	4644(77.4%)	1356(22.6%)	6000

Tab. 9: Train and Test Data Split

7.2.3 Feature Scaling

Many machine learning and optimization algorithms perform optimally when numerical features are standardized to the same scale. For instance, in Principal Component Analysis (PCA), unscaled numerical features can lead to situations where features with larger numerical ranges unduly influence the variance.

To this end, we implemented Standardization, centering features around a mean ($\mu = 0$) and scaling them to a standard deviation ($\sigma = 1$). This process aligns the features with the parameters of a standard normal distribution, thereby simplifying the task of learning weights in the machine learning models we intend to use.

$$X_{STD} = \frac{X - \mu}{\sigma} \tag{4}$$

Standardization also ensures that although the data has been rescaled we can still capture the effects of outliers.

7.2.4 Resampling

In the data set, we observed and showed in Figure 1, that we have a class imbalance; that is we have many more samples that refer to the $DEFAULT = No$ class compared to the $DEFAULT = Yes$. Blindly proceeding with this imbalance is not advised[2], as for example in our data set we could almost achieve 80% prediction accuracy of the majority class ($DEFAULT = No$), without even performing any statistical learning algorithms. There are several algorithms that can be employed to fix this, and they can be grouped as undersampling or oversampling algorithms. We give a brief overview of each class.

- Oversampling techniques create additional data points of a minority class in the aim of balancing it with the majority class.
- Undersampling techniques involve reducing the number of data from the majority class to balance it with the minority class.

We explored the oversampling technique known as the Synthetic Minority Over-sampling Technique(SMOTE) proposed by [3]. This oversampling technique generates synthetic samples rather than creating exact copies of the minority class instances. It does this by interpolating new instances between existing minority class instances. It mitigates the risk of overfitting, which is a problem with simple replication oversampling techniques, as it introduces more diversity within the minority class[3].

It is important to note that resampling methods are only performed on the training data, and the resulting training data set is only used during the training phase,

this is important because one needs to maintain a realistic testing data set to avoid deviation from the real-world scenario. After performing the SMOTE technique, we observed that we had an improved class balance in the training set as shown in Figure 18, and it's with this data set that we will proceed to train our models.

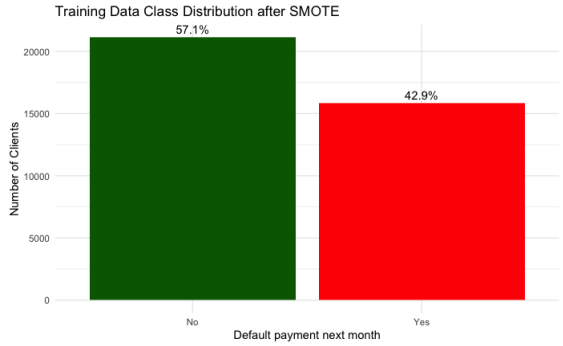

Fig. 18: Training Data Class Distribution after SMOTE