RISK ASSESSMENT
USING FEATURE EXTRACTION

Soumyae Tyagi
Data Science, Northeastern University
Boston, United States
tyagi.so@northeastern.edu

1. Abstract

In today's financial world, making accurate and fair loan/credit approval decisions is crucial for both lending institutions and borrowers. Traditional methods often rely on historical data and predetermined criteria, which might not capture the intricate patterns within the data, leading to the approval of loans that may not be repaid. In this study, the aim is to enhance the loan/credit approval process. What makes this problem so interesting is its immense practical significance in solving the issue of loan approval. Helping banks by reducing default rates and minimizing financial risks, it also ensures that individuals who are creditworthy but might not meet standard criteria have a better chance of obtaining loans. From a computational standpoint, the challenge we're addressing involves a binary classification task. Our goal is to categorize data in a manner that enables us to predict whether the provided information corresponds to an individual likely to default on a financial payment or not. n this project, we addressed the issue through a two-step approach. Firstly, we extracted features from the dataset using algorithms like PCA, autoencoders, and contrastive learning. Secondly, we applied unsupervised learning algorithms to these derived features, clustering them to make predictions about whether financial payment default will occur or not. In the second part, we implemented various state-of-the-art techniques such as GMM, K-means (treated as a clustering problem), One-Class SVM, and Isolation Forest (treated as an anomaly detection problem). The results were then compared, revealing two main findings. Firstly, features extracted from contrastive learning consistently yielded the best results across all algorithms. Secondly, K-means demonstrated the highest value of the Calinski-Harabasz Index, indicating that it formed the most optimal clusters.

2. Introduction

In today's complex financial landscape, the process of loan and credit approval stands as a critical cornerstone, impacting both lending institutions and borrowers significantly. The ability to make accurate and fair loan approval decisions is not only a matter of financial prudence but also one of economic stability and personal financial health. Traditional methods of loan approval, often reliant on historical data and predetermined criteria, have long been the standard. However, these methods face growing scrutiny as they may not effectively capture the intricate patterns within financial data, potentially leading to the approval of loans at a high risk of non-repayment.

The implications of inaccurate loan approvals are far-reaching. For financial institutions, loan defaults can lead to significant financial losses and affect overall economic health. For borrowers, unfair or inaccurate loan denials can lead to missed opportunities and financial hardship. The need for a more nuanced and accurate loan approval process is evident, and it is here that this study finds its motivation.

In the early stages of incorporating machine learning (ML) into the banking sector during the 1990s, financial institutions predominantly leaned towards supervised learning methods for predicting loan defaults. This initial preference was largely due to the structured nature of supervised learning, where algorithms learn from labeled datasets to make predictions. These models were trained on historical data where the outcomes (such as default or non-default) were already known, enabling the algorithms to identify patterns and make future predictions based on those learnings. However, as the volume and complexity of financial data increased exponentially around the 2000s, banks began to recognize the limitations of solely relying on supervised learning. This era marked a significant shift towards embracing unsupervised machine learning techniques. This shift also aligned with the evolving nature of financial markets and customer behaviors, which demanded more dynamic and adaptable models capable of handling the unpredictability and non-linearity of financial data. The utilization of unsupervised machine learning persists and continues to be relevant. And is still developing.

Traditional loan approval methods are rooted in historical data analysis, often utilizing straightforward metrics like credit scores, income levels, and debt-to-income ratios. While these methods have their merits, they often fail to adapt to the rapidly evolving financial behaviors and

diverse profiles of borrowers. This rigidity can lead to biases and a lack of adaptability in the face of complex, modern financial scenarios. The challenge lies in creating a system that is both accurate in its predictions and fair in its assessments.

The task at hand is essentially a binary classification problem: determining whether an individual is likely to default on a loan. This requires not only the accurate processing of data but also the identification and extraction of relevant features that can predict loan repayment behavior.

Our study aims to address these challenges through a novel two-step approach. We first focus on feature extraction from the dataset, employing advanced algorithms such as Principal Component Analysis (PCA), autoencoders, and contrastive learning. These methods are selected for their ability to distill and highlight the most relevant features from complex financial datasets.

The second step involves the application of unsupervised learning algorithms, including Gaussian Mixture Models (GMM) and K-means clustering, treating the problem at hand as a clustering task, as well as One-Class SVM and Isolation Forest, addressing the problem as an anomaly detection task. These techniques are utilized to cluster and identify patterns within the derived features, enabling predictions about whether a financial payment default will occur or not.

Upon comparing the results derived from the various unsupervised learning algorithms implemented in our study, two significant findings emerged, each shedding light on different aspects of the data analysis process. The first key insight pertains to the efficacy of the feature extraction methods employed. Notably, features that were extracted through the process of contrastive learning consistently outperformed those obtained via other methods across all the algorithms tested. This consistently better performance highlights the strength and effectiveness of contrastive learning in extracting features, especially for complex financial data. The second key finding was about how well the algorithms could group data. Specifically, the K-means algorithm did exceptionally well, as shown by its high score on the Calinski-Harabasz Index. This index measures how well data is grouped, with a higher score meaning clearer, better-separated groups. The high score of K-means suggests it was very effective at organizing the data into distinct groups. This is especially useful for predicting loan defaults, as it shows that K-means could accurately separate different types of data, improving our ability to predict financial defaults. These findings show that advanced machine learning methods are very effective for analyzing financial data and assessing risk. They highlight how these techniques can lead to more detailed and advanced ways of managing credit risk.

3. Methodology

3.1. We will begin by describing the algorithms used for feature extraction.

3.1.1. Autoencoders

An autoencoder is a type of neural network used for feature extraction. It learns to compress data from the input layer into a shorter representation, and then uncompress that representation back into something that closely resembles the original data. The architecture of an autoencoder has three main parts: the encoder, the latent space, and the decoder. The encoder part compresses the input data and produces the latent space, which is a more compact representation of the input data. It takes an input vector $x \in R^n$ and maps it to a hidden representation $z \in R^m$ (where typically $m<n$). The latent space is the central part of the autoencoder, representing the compressed knowledge of the input data. The decoder part then tries to reconstruct the input data from this compact space it takes the hidden representation z and maps it back to a reconstructed input $\bar{x} \in R^n$. During training, the autoencoder learns by adjusting its weights and biases to minimize the difference between the input data and the reconstructed data output by the decoder. The idea is that the compact space will retain the most significant features of the input data, essentially learning an efficient representation.

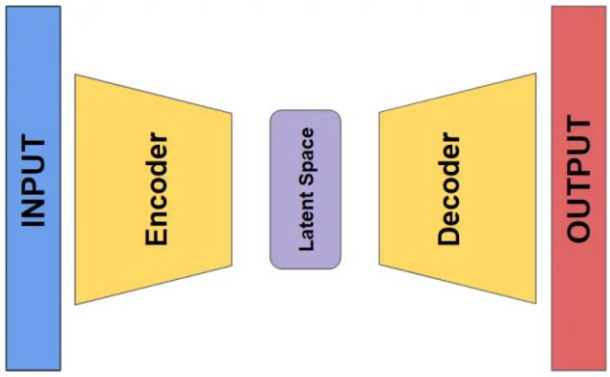

Fig. 1. Autoencoder Architecture

3.1.2. Principal Component Analysis (PCA)

It is used for reducing the dimensionality of a dataset while retaining most of the variation in given data. PCA seeks to identify the principal components, which are the directions

of maximum variance in high-dimensional data. The process begins by standardizing the data, ensuring each feature has a mean of zero and unit variance. This step is crucial because PCA is sensitive to the scale of the data. Once the data is standardized, PCA computes the covariance matrix. This matrix encapsulates how each variable in the dataset is associated with every other variable. The next step involves calculating the eigenvalues and eigenvectors of this covariance matrix. Eigenvectors represent the directions of maximum variance (the principal components), and eigenvalues signify the magnitude of variance along these directions. In simpler terms, eigenvectors define the new feature space, and eigenvalues determine their significance. The principal components are then sorted by their eigenvalues in descending order. This sorting helps in identifying which components account for most of the variance in the data. Usually, only the first few principal components are selected to represent the data, as they contain most of the important information. The final step is to transform the original data onto these new principal components, resulting in a lower-dimensional representation of the original dataset.

3.1.3. Contrastive Learning

Contrastive learning involves training a model to learn useful representations of data by maximizing the similarity between augmented views of the same instance and minimizing it for different instances. The output generated by the Contrastive Learning model consists of two different arrays of vectors, representing similar and dissimilar instances. Each vector encapsulates information about the similarity between instances. In the context of a binary classification task, these vectors can be utilized as features. In general, contrastive learning is implemented using a simple Siamese network as a base. A Siamese network is a neural network architecture designed to learn and compare similarity between input pairs. It uses the concept of shared weights that is using identical neural network parameters for multiple parallel branches, enabling the network to learn and compare similarity or dissimilarity between pairs of inputs. It usually has a few fully connected layers, and it uses Euclidean distance between dissimilar output vectors representing different clusters as contrastive loss to train the model. There is another special way to use contrastive learning, that is to use the kernel trick instead of a Siamese network to reduce dimensions and further increase the distance between dissimilar points in lower dimensions.

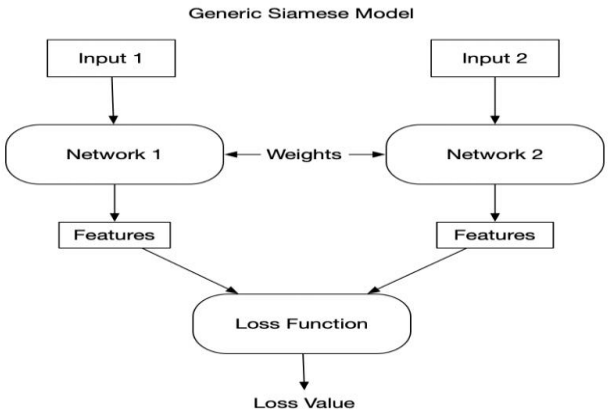

Fig. 2. Contrastive Learning Architecture

3.2. Now, let's discuss the algorithms used for anomaly detection or clustering.

3.2.1. In this project for anomaly detection, two main algorithms were used, namely, One-Class SVM and Isolation Forest.

1. One-Class SVM

The idea behind One-Class SVM is to find a hyperplane that best separates normal data from outliers. It tries to maximize the margin around normal instances. This is done by solving an optimization problem that minimizes the size of the hyperplane and the errors in classification. A kernel function is often used to help with the separation in a higher-dimensional space. The result is a model that defines a boundary around normal data, making it effective for spotting anomalies or outliers that fall outside this boundary.

2. Isolation Forest

The idea behind the Isolation Forest algorithm is to detect anomalies by using randomized partitioning, where the algorithm selects random features and split values, isolating anomalies with fewer partitions compared to normal data points. Each data point receives an anomaly score based on its average isolation path length, with shorter paths indicating higher anomaly scores. By setting a threshold, data points can be classified as anomalies or normal. This algorithm is particularly advantageous for high-dimensional datasets, as it eliminates the need for complex feature engineering.

3.2.2. For the task of clustering, we employed two primary algorithms: K-means and GMM (Gaussian Mixture Model).

1. K-means

K-means operates through an iterative process that begins with the random selection of K initial cluster centroids. In the following step, each data point is assigned to the cluster represented by its nearest centroid. The algorithm then updates the centroids by calculating the mean of all data points assigned to each cluster. This assignment-update cycle iterates until convergence criteria are met. The underlying objective of K-means is to minimize the sum of squared distances between data points and their respective cluster centroids, effectively optimizing within-cluster variance.

2. GMM (Gaussian Mixture Model).

The Gaussian Mixture Model (GMM) is a probabilistic clustering method that uses a mixture of Gaussian distributions to represent data. The expectation-maximization (EM) algorithm is used to iteratively estimate parameters for these distributions. It assigns data points, probabilities for belonging to each Gaussian component in the E-step, and it updates the Gaussian parameters in the M-step to maximize data likelihood. GMM allows for soft clustering, in which data points have a chance of belonging to multiple clusters. This method accommodates complex data patterns and is appropriate for scenarios in which clusters are not well separated.

3.3. Now, let's continue with the discussion of metrics used to assess how well the models performed: -

1. Calinski-Harabasz Index

The Calinski-Harabasz Index is a clustering evaluation metric that quantifies the ratio of between-cluster variance to within-cluster variance in a dataset. The higher the index, the better the clustering is considered.

$$Calinski\ Harabasz\ Index = \frac{Between\ Cluster\ Variance}{Within\ Cluster\ Variance} * \frac{N-K}{K-1}$$

Where N is total number of data points, and k is number of clusters.

2. Homogeneity

It measures the extent to which all elements within a cluster belong to the same true class or category. The score ranges from 0 to 1, where 1 indicates perfect homogeneity (all elements within a cluster belong to the same true class), and lower values suggest less homogeneity.

$$H = 1 - \frac{H(C \mid K)}{H(C)}$$

Where H(C|K) is the conditional entropy of true class labels given cluster assignments, and H(C) is the entropy of true class labels.

3. Completeness

Completeness is a clustering evaluation metric that measures the degree to which all elements that belong to the same true class are assigned to the same cluster. The score ranges from 0 to 1, where 1 indicates perfect completeness (all elements from the same true class are assigned to the same cluster), and lower values suggest less completeness.

$$C = 1 - \frac{H(K \mid C)}{H(K)}$$

Where H(K|C) is the conditional entropy of cluster assignments given true class labels, and H(K) is the entropy of cluster assignments.

4. V-Measure

It combines both homogeneity and completeness to provide a balanced measure of clustering quality. It ranges from 0 to 1, with 1 indicating perfect clustering balance between homogeneity and completeness.

$$V = 2 \cdot \frac{H \cdot C}{H + C}$$

Where H is the homogeneity score and C is the completeness score.

5. False positive rate

The false positive rate is the proportion of actual negative instances incorrectly predicted as positive in a binary classification problem. In this project, the goal is to decrease the false positive (FP) rate, as in the banking sector, it represents financial losses. Meanwhile, the false negative (FN) rate represents the loss of potential business for banks. Currently, the industry aims to reduce the false positive rate, even if it leads to an increase in the false negative rate, ensuring, at the very least, the avoidance of financial losses.

Where 1 represents defaulters and 0 represents those who pay back.

Actual / Predicted	0	1

0	True Negative	False Positive
1	False Negative	True Positive

4. Proposed Methodology

4.1. About Data

Heterogeneous data from multiple sources will be used to demonstrate that this approach works and generalizes well across various different datasets. The following are the primary datasets identified so far, which will be utilized in this study: -

1. Deloitte ML Challenge: Predict Loan Defaulters (Kaggle)
2. Give Me Some Credit (Kaggle)
3. fraud-detection (Data-world)

Table 1. About Data

Dataset number	Input	Expected Output
1	The dataset comprises 35 columns and 67,463 rows, pertaining to loan payment information.	The idea is to classify it into two clusters to determine whether the loan would be paid or not.
2	The dataset comprises 12 columns and 150000 rows, pertaining to credit payment information.	The idea is to categorize data into two clusters to ascertain whether credit approval is possible or not.
3	The dataset comprises 32 columns and 284807 rows, pertaining to credit payment information.	The idea is to categorize data into two clusters to determine whether credit fraud would occur or not.

Note: The data was not balanced before use, as the imbalanced distribution of classes in the dataset accurately reflects the distribution in the real world. Balancing the data may hinder the model's ability to effectively learn the patterns of the minority class.

4.2. Data preparation

Table 2. Key points on datasets.

Dataset number	Key notes
1	Dataset contains null values. Contains both numerical and categorical data. Highly imbalanced data with a ratio of 10 to 1.
2	Data contains null values. Data contains duplicate values. Highly imbalanced data with a ratio of 13 to 1.
3	Highly imbalanced data with a ratio of 577 to 1.

Data pre-processing is a crucial step before applying any clustering techniques. For this study, as a part of preprocessing, we have addressed issues like missing data, duplicate data, and handling categorical data. Since the number of missing records was not too significant, we ended up deleting records with missing data. Duplicate values were handled in a similar manner, that is, by removing the records with duplicate data. For handling categorical data, we used one-hot encoding and labeling methods to resolve this issue.

4.3. Proposed Pipeline

The suggested methodology details all the techniques used to cluster data. An overview of our approach can be observed in Fig. 3. Initially, we extracted features from datasets using techniques such as Autoencoder, PCA, and contrastive learning. We then utilized these extracted features to address the problem as a clustering task, an anomaly detection task, or a semi-supervised ML task. In the end, we compared the results and reported the best feature extraction method and algorithm for clustering or anomaly detection.

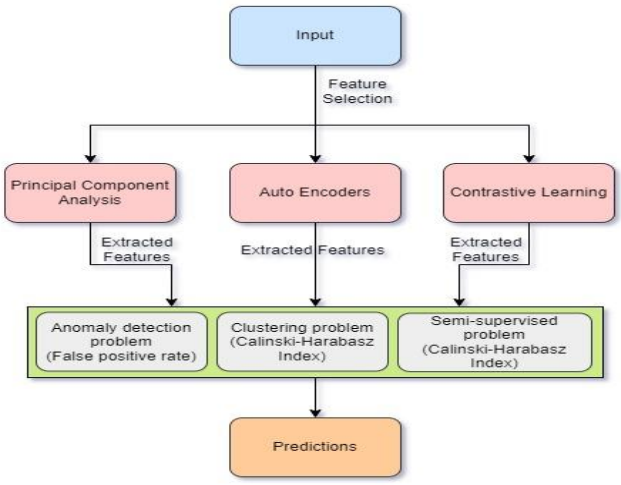

Fig. 3. Proposed Model Architecture

5. Empirical Results

5.1. Results for Dataset named: - Deloitte ML Challenge: Predict Loan Defaulters (Kaggle)

Table 3.1. For Features Extracted with Autoencoder

Algorithm / Metrics	Calinski Index	Homogeneity	Completeness	V-Measure
K-means	24761.90	4.489 e-06	2.10 e-06	2.86 e-06
GMM	2544.11	5.25 e-07	2.44 e-07	3.34 e-07

Table 3.2. For Features Extracted with PCA

Algorithm / Metrics	Calinski Index	Homogeneity	Completeness	V-Measure
K-means	4453.15	0.00011	0.0002	0.00014
GMM	2717.29	0.00017	0.00012	0.00014

Table 3.3. For Features Extracted with Contrastive Learning

Algorithm / Metrics	Calinski Index	Homogeneity	Completeness	V-Measure
K-means	11207.56	0.0001	5.24 e-05	7.25 e-05
GMM	968.59	3.32 e-06	1.60 e-06	2.16 e-06

Table 3.4. While treating Problem as an anomaly detection task. (FP rate)

Feature Extraction Method / Algorithm	Isolation Forest	One-Class SVM
Autoencoder Features	5177	303
PCA	4022	367
Contrastive Learning Features	5109	290
Without Feature Selection	2421	419

5.2. Results for Dataset named: - Give Me Some Credit (Kaggle)

Table 4.1. For Features Extracted with Autoencoder

Algorithm / Metrics	Calinski Index	Homogeneity	Completeness	V-Measure
K-means	57376.39	0.004	0.109	0.007
GMM	7681.18	0.009	0.014	0.011

Table 4.2. For Features Extracted with PCA

Algorithm / Metrics	Calinski Index	Homogeneity	Completeness	V-Measure
K-means	61362.92	0.0041	0.11	0.008
GMM	61362.92	0.004	0.11	0.008

Table 4.3. For Features Extracted with Contrastive Learning.

Algorithm / Metrics	Calinski Index	Homogeneity	Completeness	V-Measure
K-means	180248.88	0.13	0.15	0.14
GMM	97834.16	0.15	0.09	0.11

Table 4.4. While treating Problem as an anomaly detection task. (FP rate)

Feature Extraction Method / Algorithm	Isolation Forest	One-Class SVM
Autoencoder Features	13148	467
PCA	6110	510
Contrastive Learning Features	10161	365
Without Feature Selection	5519	501

5.3. Results for Dataset named: - fraud-detection (Data-world)

Table 5.1. For Features Extracted with Autoencoder

Algorithm / Metrics	Calinski Index	Homogeneity	Completeness	V-Measure
K-means	70191.20	0.44	0.13	0.21
GMM	16680.44	0.17	0.004	0.008

Table 5.2. For Features Extracted with PCA.

Algorithm / Metrics	Calinski Index	Homogeneity	Completeness	V-Measure
K-means	11845.23	0.001	0.0001	0.0002
GMM	8843.72	0.02	0.0005	0.001

Table 5.3. For Features Extracted with Contrastive Learning

Algorithm / Metrics	Calinski Index	Homogeneity	Completeness	V-Measure
K-means	604083.09	0.64	0.73	0.68
GMM	573126.02	0.68	0.71	0.69

Table 5.4. While treating Problem as an anomaly detection task. (FP rate)

Feature Extraction Method / Algorithm	Isolation Forest	One-Class SVM
Autoencoder Features	17712	1181
PCA	12178	1592
Contrastive Learning Features	81	284306
Without Feature Selection	9686	1597

From the conducted research, it can be clearly noted that features extracted using contrastive learning performed the best. This signifies that contrastive learning was clearly able to separate dissimilar records and bring closer the ones that were similar. When treating the problem as a classification task, features extracted from the autoencoder outperformed those from contrastive learning for the Deloitte ML Challenge; the reason for this could be its higher prevalence of sparse categorical data. In general, K-means outperformed other algorithms for every dataset, suggesting its ability to create well-defined clusters. This is in contrast to when treating the problem as an anomaly detection task, where features extracted from contrastive learning using One-Class SVM outperformed Isolation Forest, except for the Data-world dataset, where Isolation Forest outperformed One-Class SVM by far.

Note: As discussed earlier, the results clearly indicate that homogeneity, V-measure, and completeness may not effectively discern which algorithm performs better. This limitation can be attributed to various factors, including irregular cluster shapes, overlapping clusters, highly unbalanced data (which is the case for us), and the presence of non-numeric data. Due to these reasons, we chose the Calinski-Harabasz Index as the metric for determining which algorithm forms better clusters.

Following this, semi-supervised machine learning was applied to features extracted through the contrastive learning algorithm, using K-means as the clustering method and Logistic Regression as the prediction model. Initially, ten percent of the labeled dataset was used for the semi-supervised task. It was found that employing semi-supervised machine learning backfired, resulting in worse results. However, an increase in labeled data used for the semi-supervised task showed an increasing trend in the Calinski-Harabasz Index. At a certain point, using a very high amount of labeled data, the Calinski-Harabasz Index improved even better than the results obtained without using semi-supervised learning. It is important to note, that using a high amount of labeled data is not a good idea, as it does not accurately replicate the real-world scenario where there is more unlabeled data than labeled data.

Note: For algorithms that require specifying the initial number of clusters to start with, the initial value was set to 2. This decision was based on testing, which showed that increasing the number of clusters further decreased external measures for Clustering Evaluation. In the process, the model became more complex, leading to a loss of general interpretability.

To generalize, the majority of the best results were obtained using features extracted from the contrastive learning algorithm, treating the problem as a clustering task, and solving it using the K-means algorithm.

Conclusion and future work

To sum it up, our research highlighted that the most successful results came from features extracted through the contrastive learning algorithm, treating the problem as a clustering task, and employing the K-means algorithm. This approach not only showcased robust performance but also provided valuable insights into how it stands out as a promising methodology for applications demanding effective clustering solutions. However, certain algorithms remained unexplored due to technical constraints. Like, spectral clustering was precluded from investigation as it consistently exceeded six hours for a single run, primarily due to the high-dimensional nature of the data. Additionally, hierarchical clustering was omitted due to space limitations.

Future work involves optimizing spectral clustering for high-dimensional data, exploring hierarchical clustering, and investigating diverse kernels for contrastive learning.

GitHub Link: - https://github.com/SoumyaeCodes/RISK-ASSESSMENT-USING-FEATURE-EXTRACTION