AMSTERDAM UNIVERSITY COLLEGE

BACHELOR OF SCIENCE

CAPSTONE

Investigating general mixture models for data augmentation of niche market datasets

May 30th, 2022

Word Count: 7560

Author
Ilai BACHRACH
ilaibachrach@gmail.com
AUC

Supervisor
Breanndán Ó NUALLÁIN
o@uva.nl
AUC

Reader
Dr. Edoardo ALATI
Alati@diag.uniromal.it
La Sapienza University of Rome

Tutor
Dr. Lorenzo GALEOTTI
l.galeotti@uva.nl
AUC

Abstract

A shortage of data is one of the greatest obstacles when training supervised models. Working with small datasets is problematic because we cannot typically make precise predictions with them. A solution to this is data augmentation. In this paper, we propose a novel data augmentation method that extracts points from a larger dataset. The proposed architecture connects pythons' data fitter tool with a general mixture model (MGM). Experiments show that training a binary classifier with augmented datasets based on the MGM method provides more robust results than doing so with existing state-of-the-art methods like gaussian mixture models (GMM).

List of abbreviations -- GMMs (Gaussian Mixture Models), MGMs (General Mixture Models)

Keywords – Niche Markets, Unsupervised learning, Data Augmentation, GMMs, MGMs.

Contents

Chapter 1

Introduction

Not all data is big data. In subjects such as medicine, psychology, geology, etc., small niche datasets are the rule, not the exception. Companies working in niche markets frequently possess small amounts of data. If these companies want to use their data to make predictions, their models may not produce results with sufficient accuracy or precision. In machine learning, the reason why models perform poorly with too little data can be broken up into three main categories as depicted in this figure (Maheswari, 2019).

With a more balanced set of target columns, the model would likely Implications of less data (Maheswari, 2019)

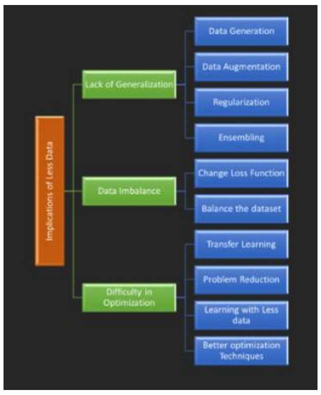
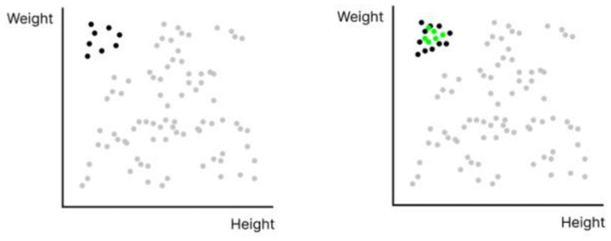


Figure 1 outlines general solutions (in blue) to a dataset that is said to have too little data. Generally, this is because models that are trained on this dataset do not perform well because the dataset is either not general enough, imbalanced, or cannot be optimized for a specific prediction task. In this investigation, we assume that the main problem with the datasets owned by companies working in niche markets is a 'lack of generalization' for their market. This is because, as seen in figure 2, a company may possess data for a niche market, as seen on the left (in black), a sub-domain of a larger market (in gray), whereas the actual niche market consists of both the black and green points (on the right).

Figure 2: A more fully captured niche market.



The small datasets owned by these companies may mislead models to learn that the entire niche market consists of their dataset, also known as overfitting. As seen in figure 1, several methods like data generation exist to tackle this lack of generalization of a dataset. This investigation makes use of data augmentation techniques to move from the left representation of the niche market in figure 2, to the right one, in which the market is more fully captured. Data augmentation is a collection of techniques for expanding and enriching a dataset while preserving its labels (Zhang et al., 2021). The central data augmentation method of this paper consists of extracting points from a large, more general dataset that are plausible additions to the company's niche market dataset. Plausible points are points that lie within the space of the company's niche market data, such as the green points in figure 2. For example, a company like Machine Learning Programs (MLP) newco B.V., possesses a large dataset it could use to extract relevant entries and to add those entries to a dataset owned by one of its partner companies working in a niche market. Through this process, MLP aims to help these companies make more precise predictions about their clients.

To identify which points from a larger dataset can be considered plausible, we first need to model a space that represents the company's niche market dataset as accurately as possible. To model a dataset, we use representation learning, which refers to a combination of techniques that allow a computer to automatically discover the representations required for feature detection or classification from raw data. (Bengio et al. 2012). Representation learning is a critical problem in machine learning since the effectiveness of machine learning is contingent on the quality of the representation (Lee et al., 2020).

A suitable representation of the niche market multivariate dataset can be obtained via a probabilistic model which assigns probabilities to points in the large dataset. A useful representation of a probabilistic model is often one that reflects the posterior distribution of the underlying explanatory elements for the observed input (Bengio et al. 2012). This probability allows us to identify which data points are "plausible" additions to the niche market dataset. Plausible points are those that have a high probability of belonging to the niche market dataset. This means that this plausible point can be added to the niche market dataset without particularly changing its underlying representation (Arora et al., 2021).

Probability distributions like Gaussians are particularly useful in the context of understanding the likelihood that a given data point is in a distribution. This is because when we exploit the Central Limit Theorem, we can assume that any large enough sample can be approximated by a Gaussian. For this reason, we can also use multiple Gaussian distributions to describe our niche market dataset. Gaussian Mixture models (GMMs) are probabilistic models in which all data points are assumed to be produced by a sample of Gaussian distributions with unknown parameters (Pan et al., 2020). However, this is an approximation. Certain features in a given dataset can be distributed in an exponential fashion. Other features may contain outliers that are best modeled using a Poisson distribution that can describe rare events. When modelling a combination of features using only Gaussian, information that was not captured by the gaussian representations of features is potentially lost. This paper proposes the use of a model that incorporates this seemingly lost information in the form of General Mixture models (MGM).

The objective of this paper is thus summarized as: *By training a final classifier on an augmented dataset that includes points extracted from a large dataset by a general mixture model that represents our niche market dataset, we aim to improve our classifiers' performance over an augmented dataset with points extracted using a gaussian mixture model.* For this, we make use of the **Home Credit Default Risk'** dataset from Kaggle. While this dataset is open and free to use, most of its features have fewer than 3 unique values. However, for the purpose of this study, this dataset allows us to create both a large as well as a niche market dataset. We can then see whether the MGM-based representation of our dataset lets us add extracted points to our niche market dataset to improve the results of a model trained on this augmented dataset compared to the GMM-based representation.

Chapter 2

Related Works

One of the most difficult aspects of training supervised models is the paucity of labeled data for training and the resulting overfitting and underfitting issues (Arora et al., 2021). Data augmentation is one of the answers to this issue. In computer vision and image processing, data augmentation has come a long way. Many advances have been made in data augmentation of picture files, particularly in medical image datasets, by making modifications to the original file, such as random cropping, flipping, rotating, and so on, to create a new sample file. As a result, there have been several advancements in medical science. Bowles et al., for example, suggested using Generative Adversarial Networks (GANs) to extract additional information from a medical dataset by producing synthetic samples that looked remarkably similar to the genuine photos. GANs are particularly useful with image data (Bowles et al., 2018).

In comparison, advancements with numerical datasets are limited. Arora et al. (2021) propose employing Gaussian Mixture Models (GMMs) to generate data that is quite comparable to the original numerical dataset. The findings showed that the Mean Absolute Error decreased, indicating that the regression model improved in accuracy (Arora et al., 2021). In several other studies, GMMs have shown limited success in the domain of data augmentation (Pan et al., 2020, Nguyen, 2011).

Several other techniques that model parametric and non-parametric distributions have been proposed to improve multivariate data representation. The research community has suggested improving GMMs by investigating the identification of the number of components for initializing the GMM. Wan et al. 2019 proposed SC-GMM, which rapidly determines the appropriate number of gaussian components for each class and then classifies a fresh sample using these gaussian components. Another method known as i-Mix is used to improve contrastive representation learning (Lee et al., 2020). i-Mix is especially successful when the size of the training dataset is minimal or when data augmentation is not possible in other ways. In contrast, when datasets are large, manifold learning, which uses a non-linear approach, can reduce dimensionality. Manifold learning approaches (e.g., t-SNE, UMAP) offer the benefit of retaining not only the topology but also the distance in the higher dimensional space (Hemmati et al., 2019). Frequently, these papers focus on the generation of data and do not make use of larger data sources for data augmentation purposes.

This study proposes the use of large and perhaps publicly available datasets for the finding of new points. Further, unlike GMMs, general mixture models (referred to as MGMs to avoid confusion) are a type of unsupervised probabilistic model made up of multiple distributions called components and their corresponding weights. This enables more complex distributions to be modelled in relation to a single underlying phenomenon. Unlike GMMs, MGMs can, in theory, incorporate the shape of the distribution of individual features. As previously outlined, certain features are more accurately represented by non-gaussian distributions. This can also be extended to parts of features that may belong to a gaussian cluster. These representations can be fed into a General Mixture Model that makes use of this information even in higher dimensions. MGMs can also be used to study the shape of the distributions of binary features, which can be modeled with Bernoulli distributions (Schreiber et al., 2018).

The landscape of solutions that are part of a pipeline that incorporates univariate representations and improves the performance of data augmentation procedures is manifold. Unlike other methods outlined above, MGMs specifically focus on integrating more targeted distributions that more accurately capture the shape of a set of points belonging to a feature into a combined model. This paper explores MGMs' role in improving the fitting process of multivariate data for the task of data augmentation. By capturing the shape of parts of a numerical niche market dataset more precisely than a GMM alone, the model can, in some instances, capture the entire dataset more precisely. The MGM can therefore act as an addition to the current state-of-the-art GMM-based modelling approach.

Chapter 3

Preliminary Concepts

Gaussian Mixture Models

GMM is a clustering technique where each cluster is modeled using a Gaussian distribution. Complex assignments, rather than rigid cluster assignments like k-means, are the result of this flexible and probabilistic approach to data modeling (Brownlee, 2019). Each component has a probability of representing a given data point.

A Gaussian Mixture Model Density estimation entails choosing a probability distribution function and its parameters that best describe the observed data's joint probability distribution. According to the equation, a Gaussian mixture model is a weighted sum of M component Gaussian densities.

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{M} w_i \; g(\mathbf{x}|\mu_i, \; \mathbf{\Sigma}_i),$$

The parameters of the Gaussian Mixture Model are then fitted using the expectation maximization algorithm (Reynolds, 2009). In the first expectation-step, the posterior distribution is calculated based on the existing parameters. This is to determine which Gaussian is accountable for which data point. In the maximization step we use the probabilities collected in the E-step to determine the updated parameters (μ, Σ). These two steps are repeated until convergence. In the expectation phase, the latent variables are summed together. The fundamental function of latent variables is to enable the representation of a complex distribution over the observable variables using a model generated from comparatively simple (usually exponential) conditional distributions (Bhattacharyya, 2020). Like this, the model creates clusters for points that share a similar location in space. The model's goal is hence to capture most of these points (Brownlee, 2019).

It should be noted that GMMs, unlike distance methods like K-means, are particularly useful in this context because they assign a probability to points. This probability is the likelihood that a given point falls within the distribution we are modelling.

We also make use of a Bayesian GMM. For the weight's distribution, this class implements two kinds of priors: a finite mixture model with Dirichlet distribution and an infinite mixture model with the Dirichlet Process. In practice, the Dirichlet process is estimated by using a truncated distribution with a set maximum number of components. The exact number of components utilized is nearly always determined by the data (Scikit learn Bayesian GMM, n.d.).

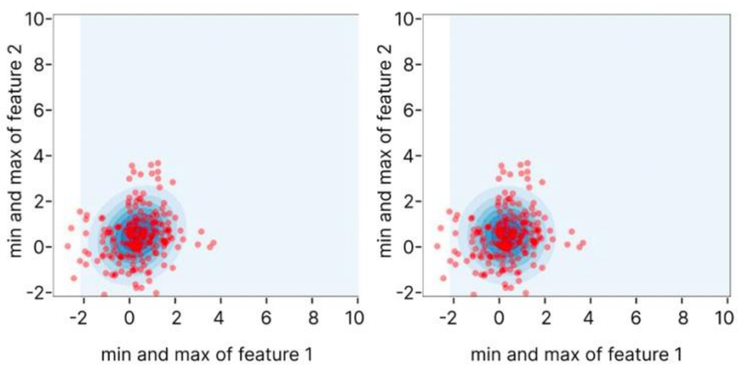
General Mixture Models

A mixture model is made up of probability densities D1,..., Dk and mixing weights or proportions w1,..., wk, where k is the number of component distributions (Pan et al., 2020).

$$P(x \mid D_1, \ldots, D_k, w_1, \ldots, w_k) = \sum_{j=1}^{k} w_j P(x \mid D_j)$$

When modelling an underlying dataset, we can therefore use a combination of different probabilistic distributions with a certain weight. Within the model, these distributions are fit inside independent components where each component represents a cluster as identified by the GMM. To illustrate this further, the following diagram shows a one-cluster fitting of a two-dimensional generated dataset.

Figure 4: An example of a gaussian and a general mixture model capturing a two-dimensional generated dataset



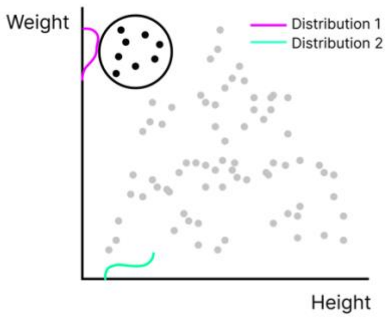
On the left a Gaussian Mixture Model is presented. On the right side a General Mixture Model is presented.

By incorporating multiple probabilistic distributions, the aim is to capture the dataset more entirely. Like a GMM, an MGM uses expectation maximization for parameter optimization.

Different distributions

The General Mixture Model will take in a collection of distributions. Univariate distributions require a distribution per dimension. Examples include the uniform distribution for two values, or the exponential distribution. We can configure so-called independent components within the MGM. Each independent component is made up of a set of points from each feature. Each set of points can be modeled with a different univariate distribution, as shown in Figure 3.

Figure 3: Diagram of a 2D setting to show how a certain cluster represented by the black circle, contains points that are part of 'Weight' and 'Height' of a person.



These two sets of points may have 2 completely different distributions, as outlined in purple and green. The MGM aims to incorporate these shapes. This process is scalable to N dimensions. As part of the MGM, we can also choose to fit multivariate distributions such as multivariate gaussian distributions. This is also known as a joint normal distribution, which is an extension of the univariate normal distribution to more dimensions. A random vector is n-variate normally distributed if each linear combination of its n components possesses a univariate normal distribution. We can then use a combination of multivariate and univariate distributions to fit our MGM (Schreiber et al., 2018).

Light GBM

The evaluation of this investigation will be conducted on the Light GBM, which will be trained on the augmented MGM and GMM datasets, respectively. Light GBM is a fast, distributed, and high-performance gradient boosting architecture based on the decision tree approach. Its main applications are ranking and classification (Khandelwal, 2017).

Chapter 4

Methodology

We created a new algorithm that can segment a dataset into clusters by fitting a GMM, then uses fitter functions to create distributions corresponding to elements of each cluster and automatically creates independent components which are fit inside a general mixture model. We combine python's fitter library and pomegranate library in a new library that is projected to be released end of August. **The algorithm outputs a new dataframe based on a smaller client dataset S and a larger dataset G with new extracted points added to the dataset S.** Whether this new dataset helps a model at an evaluation task is then tested using different classifiers and settings.

In this investigation, we follow the data science methodology:

- **Tools**: Initial Experiments with libraries (See Appendix A for an overview of the used libraries)
- **Preprocessing**: Data preparation will be covered in Chapter 5.
- **New Approach:** Automated creation of an MGM
- **Training & Modelling:** Training the LGBM classifier with an augmented dataset based on the MGM and comparing this to a LGBM classifier trained on an augmented GMM-based dataset. This will be covered in Chapter 5.
- **Evaluation:** Looking at results of a *binary classification task* to see if our new MGM method has improved accuracy and precision in comparison to the GMM based dataset. This will be covered in Chapter 5.

This work was carried out at Machine Learning Programs in the four-month period of February through May 2022.

New Approach: The MGM pipeline

The MGM pipeline consists of the following formalized steps:

a. *Training and assigning clusters to each row of the dataset.*

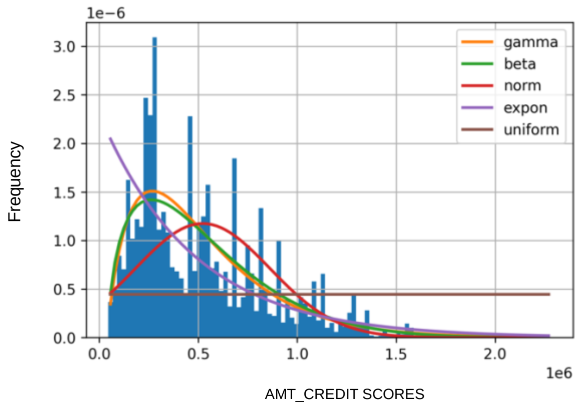
This is done using either the GMM or the Bayesian GMM from Sci-kit learn. Unlike the simple GMM, the Bayesian GMM uses the optimal number of components given a maximum number. See chapter 3 for an explanation of how these models work.

b. *We fit and return the best distribution including their parameters in* ***pomegranate*** *format for a given set of points using python's* ***fitter*** *library. Here we are connecting the two libraries. There are around 7 distributions that overlap in both libraries.*

These 7 distributions are as follows: Normal distribution, exponential distribution, gamma distribution, lognormal distribution, uniform distribution, beta distribution.

For a given feature we can model the underlying distribution as seen in Figure 4.

Figure 4: Modelling the distribution of AMT Credit



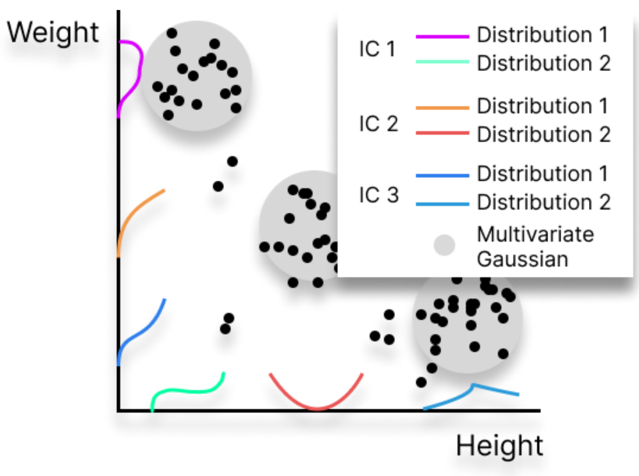
Unlike Pomegranate, Fitter can return the best fitted distribution given a set of them. It also contains many more distributions than are available in the pomegranate library. Therefore, we must find the overlap of distributions available in both libraries if we want to use them together. It is recommended that pomegranate implement support for all of python's fitter libraries to optimize this process. To choose the 'best' fitted curve we use the sum-square error measure, this can be changed depending on the situation.

As can be seen in figure 4, not all distributions follow the same trend. By selecting the best one, in this case the gamma distribution, we aim to capture the part of features that are part of a component more precisely.

c. *We repeat step 2 for all sets of points for each cluster. For each cluster we also initialize a multivariate gaussian.*

If we are conducting c with 2 features and 3 clusters our model is constructed as follows.

Figure 9: MGM Model construction.



We assign a multivariate gaussian and an independent component (IC) to each cluster. The IC incorporates the shape of the distribution respective to each feature.

d. *We create the General Mixture Model by placing the distributions for a given cluster into an independent component and adding the initialized multivariate gaussians for each cluster.*

Like the GMM, this MGM can be used to assign probabilities to new points. Given our MGM's representation of our S dataset, we can now assign log probabilities to **new** points from a G dataset. The domain of log probabilities is $(-\infty,0]$. Higher log probabilities indicate the model is determining points to lie within the initialized distribution. The MGM assigns probabilities to each component of the mixture. For example, for a given 2 dimensions, 3 clusters, the MGM will output 6 probabilities. One for each independent component and one for each multivariate gaussian. We chose the maximum value for every point.

The MGM therefore takes in the following elements with a choice of N clusters and M Dimensions:

> *MultivariateGaussian**1**(Comp0)**…**MultivariateGaussian**N**(Comp(**N**),*
> *IndependentComponent**1**(Comp0 Distribution(**1**)**…** Comp0 Distribution(**M**))**…***
> *IndependentComponent**(N-1)**(CompN Distribution(**1**)**…**CompN Distribution(**M**)))*

As demonstrated in diagram 9, each component is fitted to the set of points belonging to each feature. Note that the number of dimensions of the multivariate gaussian, and the *number of distributions* we place into the IndependentComponent are dependent on the number of features we fit the MGM with. We only run the MGM with at least 2 components and 2 features.

We can then choose points from G that have a probability higher than a certain threshold and add those points to S.

e. *We use two distinct thresholding approaches to determine which points to add to S*
   1. The threshold is computed based on G and a certain percentage quantile of the highest points. This is a more general approach as it does not consider S in finding the threshold.

      For the MGM we select the highest probability out of the given probabilities of a point belonging to one of the components. The threshold is then computed based on a percentage value that is used to determine the quantile. All values greater than this value are augmented to the niche dataset.

   2. The threshold is computed based on S and the 80th percentile of the highest points. This is a more specific approach as the threshold is computed based on probabilities assigned to S which the model was trained on.

      Given the maximum probabilities for the MGM for the niche market dataset itself, we order the values and then compute a threshold index using a given percentage. We augment all values above this index. Since we determine the threshold based on the same dataset that is used to initialize the MGM, this threshold is likely closer to the center of the niche market dataset.

Note that we can use the GMM from step a to output probabilities too and create a new S dataset with extracted rows from G in the same way. We use this as our baseline model.

Chapter 5

Implementation Details

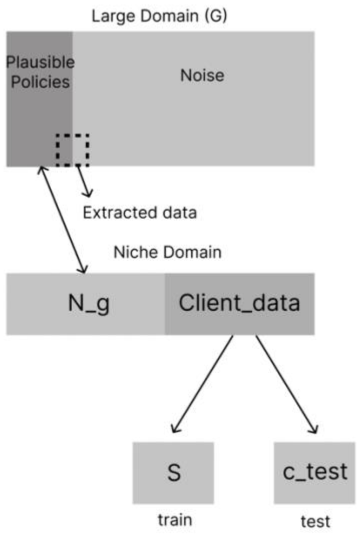
Dataset

In this investigation, we make use of the '**Home Credit Default Risk'** dataset from Kaggle. This dataset contains more than *300,000 entries with ~120 features*. This dataset is open and free and can be used to manually create a niche market dataset. *Home Credit uses a range of alternative data, such as telecom and transactional information, to estimate their customers' repayment ability to ensure that their underserved demographic has a favorable lending experience.* The dataset can be found here.

The output column of the dataset is a binary target of either zero or one. A one represents the client has repayment ability. This models the assignment posed within the MLP company. The target column is highly imbalanced. *230276 instances = 0 and 21832 instances = 1*. Furthermore, *around ~50 of the columns are binary features*. This provides a hurdle as the MGM, while it supports a Bernoulli distribution for binary variables, was not primarily tested on this. Additionally, several features maintain negative values.

Using this dataset, we segment it into a larger domain G and a smaller niche domain. See figure 10.

Figure 10: A representation of how we split our Home Credit Default Risk Dataset



Given our large dataset we create a niche domain. This **niche domain** consists of a particular set of individuals that share commonality such as being in the same age range. For this investigation we distinguish between a hard-cut niche that is more easily distinguishable and a complex or complex niche in which individuals share attributes that may not be visible immediately.

Hard Niche

We first create a hard niche domain based on the following characteristics for entries:

CNT_CHILDREN= the number of children equals 0
AMT_INCOME_TOTAL= limiting the total income to 147150
NAME_FAMILY_STATUS= only Married families
AYS_BIRTH= > 27 years (10000 days)

We then split this domain into two parts (40 60 split), one of which consists of points (N_g) that we aim to find in the large domain G and are present there from the start. The other part, the client_data, is what is available for modelling purposes. We train our GMM and MGM on the S dataset and aim to augment this S with as many points from N_G as possible. Hence our goal is to extract as many points as possible from G that are in N_G, which we name 'plausible policies'. We then use the c_test dataset to test them. The resulting sizes of the datasets are:

G: | 283382 | 124 |
S: | 12064 | 124 |
C_test: | 12065 | 124 |
N_G: | 36195 | 124 |

Complex Niche

To create the more complex niche, we use a gaussian that does not select points based on a hard cut interval like the hard niche. First, we select a mean and a variance, so we can choose a range for a given feature or collection of features and input this into the diagonal of the covariance matrix. We chose the following features as they are not the most important based on the feature selection scores and can therefore have a more subtle effect on the niche we are selecting:

A - AMT_CREDIT
B - AMT_ANNUITY
C - DAYS_BIRTH
D - DAYS_EMPLOYED
E - AMT_INCOME_TOTAL

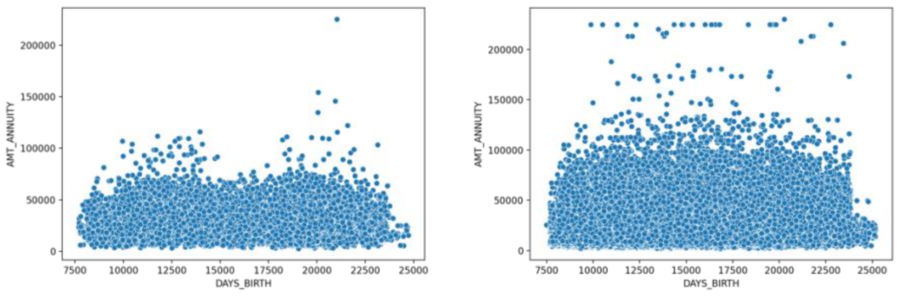
We make several gaussian selections of different combinations of these features, and then remove duplicates. It is important to note that for the chosen combinations of features, we create a random multivariate gaussian using the set means and covariances. This allows us to select points from a complex edge. The following shows the combinations for which we apply a gaussian and select points.

Combo 1 = A, B → Mean = (150000, 50000) - Co-Var Matrix = [200000^2, 0] [0, 10000^2]
Combo 2 = A, B → Mean = (500000, 20000) - Co-Var Matrix = [200000^2, 0] [0, 10000^2]
Combo 2 = C, A → Mean = (21000, 100000) - Co-Var Matrix = [1000^2, 0] [0, 800000^2]
Combo 3 = C, E, → Mean = (12000, 1000000) - Co-Var Matrix = [1000^2, 0] [0, 100000^2]
Combo 4 = B, E → Mean = (30000, 100000) - Co-Var Matrix = [1000^2, 0] [0, 100000^2]

As stated, we next remove duplicates to grab the intersection of these points. We then split the data into the different datasets like in the last step we did for the hard niche. Finally, we are left with a more realistic and complex niche of combinations of ranges of these features that are not visible at first sight (Hamberger, 2022).

See below for an illustration of the resulting complex niche using 2 features.

Figure 10b: The selected complex niche



Left: Complex Niche, Right: Entire G domain

As shown in figure 10b, the selected complex niche does not have a hard cut out like the simple niche described before.

Preprocessing data for the MGM

To receive optimal results, we remove NaN values in the existing dataset and replace those with the mean of the feature. We also exclusively select features which are of type bool, int or float. Due to the complexity of the methodology outlined in section 4 we also run a method in SCI-KIT learn known as recursive feature selection to determine the most important 7 features and select those to run experiments. We also apply an absolute value to the features which have negative values.

Preprocessing for the LGBM

To run the LGBM classifier for each dataset, we need to add the extracted points from our MGM method to the 's' dataset we use for training.

Model Set-up and training details

In this investigation, we are using an LGBM classifier. The training technique is sped up by bucketing continuous feature data into discrete bins. This allows for faster training speed and higher efficiency. We run the LGBM classifier with *20 jobs and verbose set to 0.* We train on the dataset S after dropping the target column. We train two LGBMS, one for the baseline GMM-based dataset and one for the MGM-based dataset. When running the LGBM with N clusters, we turn off the Bayesian method. We also do not change the weights of components within the MGM. This can be set if desired.

Chapter 6

Experiments and Results

Metrics

Precision Score. The main metric used is the *precision score* of the model. The precision is the ratio of tp / (tp + fp) where tp is the number of true positives and fp is the number of false positives. We use this to determine the classifier's capacity to avoid labeling as positive a sample that is negative (Scikit learn Precision, n.d.).

Matthew's correlation coefficient score (MCC). The MCC incorporates both the number of false positives and false negatives of our models' predictions to show any imbalances present in those predictions.

$$\mathrm{MCC} = \frac{\mathrm{TP} \cdot \mathrm{TN} - \mathrm{FP} \cdot \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP}) \cdot (\mathrm{TP} + \mathrm{FN}) \cdot (\mathrm{TN} + \mathrm{FP}) \cdot (\mathrm{TN} + \mathrm{FN})}}$$

We can use this to see how well the model did across the confusion matrix, as it only provides a high score if the forecast performed well in each of the confusion matrix's four areas (Chicco et al., 2020).

Brier Score. We use the Brier score to compute the mean squared error between anticipated and observed probabilities.

$$BS = \frac{1}{N} \sum_{t=1}^{N} (f_t - o_t)^2$$

Here f_t is the anticipated value, whereas o_t is the observed value. N denotes the quantity of observations. It allows measuring the accuracy of the model (Dash, 2022).

We also make use of the *percentage overlap between points extracted from G and those part of the Niche Domain (N_g)* to see how well our model is at selecting relevant points.

Finally, we analyze the *time complexity* of the two models.

Experimental Procedures

My experiments section is divided into **two** main parts. First, I tested out the pomegranate distributions related a toy dataset. In the second part, we try to extract points from the Kaggle Credit Score dataset to solve the binary classification problem.

Part 1

To understand the underlying mechanisms of python's pomegranate tool which allows us to create general mixture models, we began with 7 different experiments. For this we generate a two-dimensional dataset with 100 points of feature x that follows an exponential distribution and a feature y which follows a normal distribution.

Experiment 1. First, we fit two specific distributions, in this case one exponential and one normal distribution inside an independent component that we feed into an MGM. This does not provide any results. We then apply a GMM.

Experiment 2. Here we fit an MGM with the distributions directly.

Experiment 3. Fitting a normal distribution with a diagonal covariance matrix after the initialization of independent components within an MGM does not work.

Experiment 4. Here we create one model that takes in one multivariate gaussian and two distributions.

Experiment 5. This model takes in independent components and a multivariate gaussian initialized by random parameters.

Next, we use python's fitter library to initialize parameters for each dimension respectively.

Experiment 6. Like model 5, this one takes in two independent components, one for each feature and initializes these based on parameters from the fitter library.

Experiment 7. We fit a model without a multivariate gaussian as part of the MGM.

Part 2

We distinguish between hard niche and complex niche. For both niches we experiment with the specific and general threshold approach mentioned in the methodology part e. We do these with a 70 and 80 threshold value. Within the thresholding approach, we make sure to select only maximum values below zero, as rounding issues cause a value of "-0" to be selected in some instances. Beforehand, we select the top 7 features of S using Feature_selection with the RFR score.

Note that we plan to focus the main experimentation on the complex niche as it is a more realistic scenario to work with a complex niche like the one that was designed.

1. We follow the methodology outlined in chapter 4 for the first 7 features. This means that the first two features train our GMM and MGM model and return two *precision scores*. This is repeated for the first three, four, five, six and seven features. The order is of features ranked as most to least important from the feature selection procedure. We do not change the order here. For the hard niche: We maintain the number of components at 3. We run 3 trials and take the highest reported precision scores from all trials. Unless there is no convergence, in which case we repeat the trials.
    a. *For the complex niche, we do this step with n_components = 3, 4, 5.*

2. We follow the methodology from chapter 4 for 7 components with a set amount of the first 4 features. To reproduce these results, turn off the Bayesian approach so that each component is used as initialized.
    a. *For the complex niche, we do this step with n_features = 2, 3, 5.*

3. We check the ideal number of components based on the Bayesian information criterion (BIC) method and discuss whether those yielded the best results (Filiz, 2022).

4. We calculate the highest overlap percentage for both experiments 1 and 2, between extracted plausible entries from G and added entries to S.

Results discussion

Part 1

Experiment 1 Result. The applying process fits the data well but overrides the independent component previously initialized.

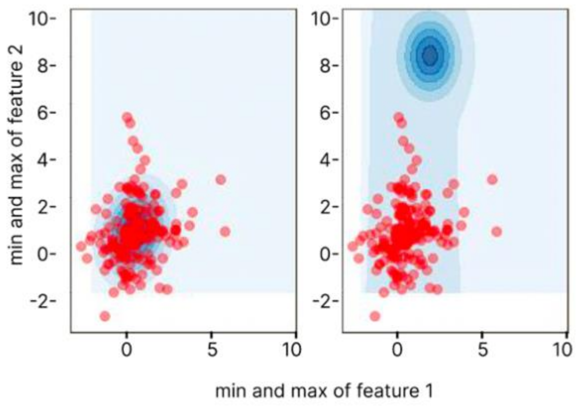
Experiment 2 Result. To use python's fitter library to initialize parameters of individual distributions, we need to fit them inside so-called independent components within the MGM. Otherwise, the MGM only takes in uninitialized distributions.

Experiment 3 Result. Like in experiment one, this new model overrides the MGM.

Experiment 4 Result. This demonstrates that the distributions must be fit within independent components rather than by themselves.

Experiment 5 Result. This approach is promising. Figure 5 shows that it produces a relevant result but with the false location in space.

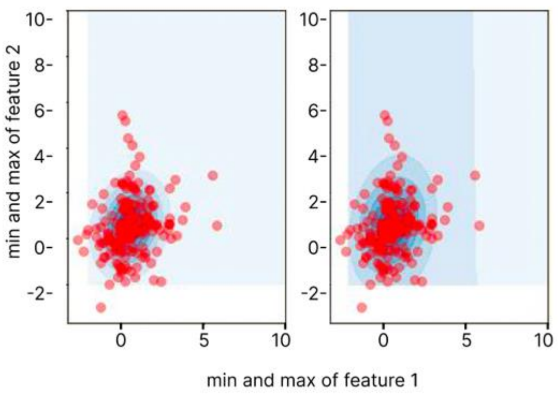
Figure 5: Experimenting with general mixture models uninitialized parameters



Left: GMM, Right: MGM

Experiment 6 Result.

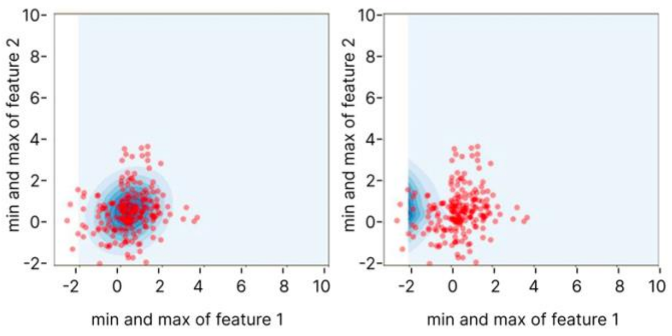
Figure 6: Experimenting with general mixture models initialized parameters



It looks like this model is capturing the dataset well.

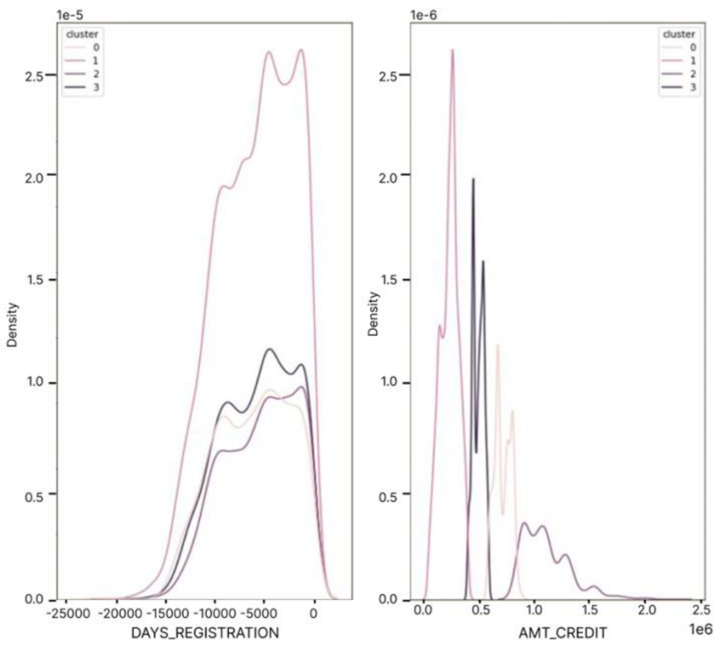
Experiment 7 Result. The independent components do not fit the data well.

Figure 7: Experimenting with general mixture models only independent components



As can be seen these independent components do not capture the data well by themselves. Improving the initialized parameters does not have an effect. It becomes evident that we need one independent component per cluster, with distributions fitted prior to initializing the MGM and a multivariate gaussian for each cluster. This becomes evident when considering the following figure for two features of the credit-score dataset used in this investigation.

Figure 8: Cluster analysis of two features in credit-score dataset



This figure shows the clusters generated from a GMM of the two features “Days Registration” and “AMT_CREDIT”. Each cluster is a specific color and includes points from each respective feature. These points belonging to each feature have different underlying distributions. In some cases, these are even well separated by the GMM as seen on the subplot on the right. However, a current GMM does not capture the shape of these distributions. The aim of the MGM is to incorporate these distributions to create a model that can capture more information of the underlying dataset. This was first discussed in figure 3.

Given the outcome of these first experiments, we created a MGM pipeline that automates the process of assigning a cluster to each row of the dataset, then fitting distributions for each set of points belonging to a cluster, and finally combining these with a multivariate gaussian for each cluster inside a general mixture model.

To incorporate the fitted univariate distributions into our final MGM we make use of so called “Independent Components”, which allow us to model each dimensions’ set of points independently. To incorporate covariance between different dimensions we then aim to add the multivariate gaussian distributions into our final mixture.

Part 2

We ran over 70 experiments for the complex niche using this novel MGM architecture. For a full list of the experiments see Appendix B. Since this is the more realistic niche, we repeat steps 1 and 2 from part 2 of the experimental setup with more combinations of features and components. The results for the LGBM model trained for a binary classification task on the augmented complex niche market dataset are summarized in the following tables. All results are rounded to 2 decimal places.

Figure 11: The top 5 results for GMM and MGM with up to 7 features.

Experiment	Model	Precision	Brier Score	MCC Score	Confusion Matrix*
Threshold: 2, cut-off: 70%, # of Features: 2 # of clusters: 4.	GMM	.67	.08	.08	11517 5 / 945 10
Threshold: 2, cut-off: 70%, # of Features: 3 # of clusters: 3.	MGM	.63	.08	.08	11515 6 / 943 12
Threshold: 2, cut-off: 70%, # of Features: 2 # of clusters: 3.	MGM	.62	.07	.08	11517 5 / 947 8
Threshold: 0, cut-off: 70%, # of Features: 2 # of clusters: 5.	MGM	.60	.08	.08	11514 8 / 943 12
Threshold: 2, cut-off: 70%, # of Features: 2 # of clusters: 5.	MGM	.59	.08	.07	11515 7 / 945 10

*The confusion matrix reads as follows from top-left: TN, FP, FN, TP.

This table is the result of selecting the top 5 precision scores from a series of experiments as conducted in part 1 of the experimental section. We vary the number of features and run an MGM and GMM respectively on each combination of threshold approach (either setting 0 or 2) and number of clusters (3, 4, and 5) as well as a 70% and 80% threshold, respectively.

Looking at the precision scores, the GMM, our baseline model, performs the best with the more specific threshold, (setting: 2), with only 2 features and 4 clusters. Otherwise, the MGM takes the next 4 highest precision scores. This trend is also visible in the hard niche experiments. For reference, see Appendix C. **The GMM seems to require a specific combination of parameters, but if these are found it performs well. In contrast, the MGM performs better on several more occasions**. For reference, note that training on the entire G domain yields a precision of 40% whereas simply training on S yields a precision score of around 25%. This suggests that training a model on a combination of the original S dataset and extracted points that are more relevant to the niche domain, improves the results of the classification task.

Reflecting on threshold setting 2, it seems like in most instances, this S-specific thresholding approach (as the threshold is determined based on S rather than G), performs better than the more general approach as outlined in the experiments section. Furthermore, the data suggests that a lower threshold that generally allows for more points (and results in a larger augmented dataset) in combination with the threshold setting 2, also improves the results. This is interesting as these augmented datasets combine both more data with a more S-specific threshold approach. Furthermore, from the various experiments that were conducted, it becomes clear that the **percentage value has a larger impact than the thresholding setting, on the resulting precision scores of the model.**

We also look at the Brier and MCC scores. In all cases, these scores hover around the same value. While a low Brier score suggests a low mean-square error, which is positive in all cases, the MCC score is relatively low as the model incorporates many false negatives. This is likely due to the imbalance in the predictions of the target column as mentioned in Chapter 4. The model hence overestimates the number of 0 targets[1]. For the hard dataset, on the other hand, the highest MCC score was around 0.11 for the 2 feature-based MGM with a 70% threshold with setting 2. This, as well as evidence from higher precision scores as seen in Appendix C suggests that the models perform generally better on the simple niche.

Next, we look at figure 12 below, where we vary the number of components as outlined in Part 2.2 of the experimental setup.

Figure 12: The top 5 results for GMM and MGM with up to 7 components

Experiment	Model	Precision	Brier Score	MCC Score	Confusion Matrix
Threshold: 0, cut-off: 70%, # of Features: 3 # of clusters: 6.	MGM	.65	0.08	0.06	11516 6 944 11
Threshold: 0, cut-off: 70%, # of Features: 3 # of clusters: 2.	MGM	.58	0.08	0.06	11516 6 947 8
Threshold: 0, cut-off: 70%, # of Features: 2 # of clusters: 6.	MGM	.56	0.08	0.07	11515 7 946 9
Threshold: 2, cut-off: 70%, # of Features: 2 # of clusters: 6.	MGM	.55	0.08	0.07	11513 9 944 11
Threshold: 2, cut-off: 70%, # of Features: 5 # of clusters: 5.	MGM	.54	0.08	0.08	11509 13 940 15

Here, the MGM outperforms the GMM in terms of precision in almost all experiments. Surprisingly, experiments with either many clusters (>= 6) or a small number of clusters (2) perform well. Like in figure 11, the 70% threshold does better. However, the threshold setting seems to play less of a role when varying the number of components. This might be because the threshold percentage as well as the number of components are more significant in capturing S accurately.

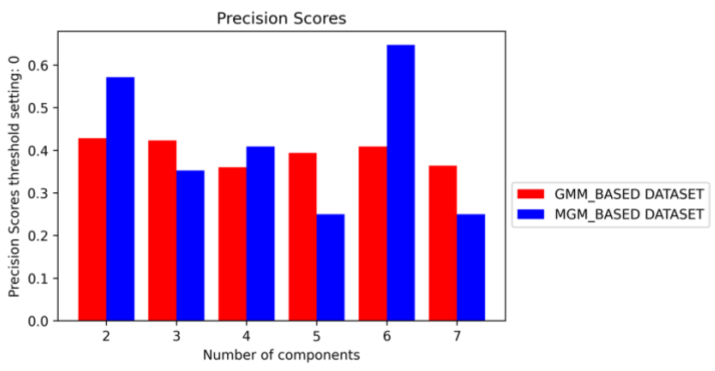
Also, note that in several cases, the MGM is unable to converge, either because it mislocates its representation of the data, does not produce any results, or does not find relevant distributions to fit. With these results, it becomes evident that fine tuning the parameters for threshold, number of features and number of components can greatly improve the precision scores for the MGM model. It can also make it yield higher precision scores than the GMM on average, making the MGM more robust than the GMM.

For step 3 of the experimental set-up, we investigate the ideal number of components for this dataset, which we base on the Bayesian Information Criterion (Filiz, 2022). We find that the

[1] The precision scores are also affected by a strong imbalance in the target column. As the number of False Positives is always low the model is likely too conservative in estimating that someone has repayment ability.

ideal number of components is between 5-7 with equal scores. As seen in figure 12, the best scores are generally obtained with 5 or 6 components. However, this is not always the case, as can be seen in Figure 13.

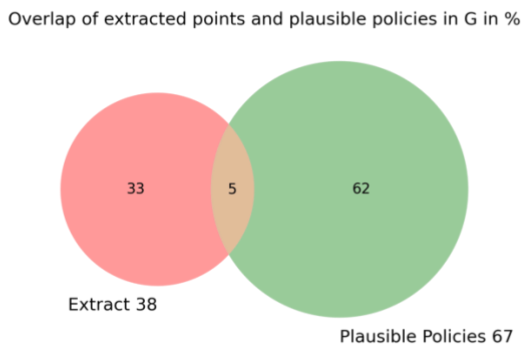
Figure 13: Precision scores are not always the best with high number of components



Like Figure 12, Figure 13 investigates varying the number of components but for a specific threshold setting of 0 with a percentage of 70%. As can be seen, **the MGM and GMM can yield relatively high precision scores with a low number of components too**. Unlike the results from Appendix C for the hard niche, for which the MGM generally does not yield results for a high number of components, for the complex niche the MGM seems to converge well.

For step 4, we look at the percentage of overlap. The following diagram illustrates this percentage overlap with an example.

Figure 14: 80% threshold percentage overlap demonstration


Overlap of extracted points and plausible policies in G in %
33
5
62
Extract 38
Plausible Policies 67

The extracted points only contain a few extracts that were part of N_G, the plausible policies we aimed to find. The highest percentage overlap for the complex niche was around 8% for the 2 feature-based MGM with 6 components, with 70% threshold setting 2 which had around 67% precision. For the simple niche we received a 10% overlap also for the same parameters which had around 76% precision. From these experiments it seems that the higher this extraction percentage, the higher the precision scores in subsequent evaluation tasks. This, however, requires further experimentation. With support for more distributions, it seems that the MGM could capture the dataset more entirely and is able to extract a larger number of plausible policies.

Essentially, we found that the GMM does better in very specific settings, otherwise the MGM has high precision scores in most experiments. This versality is likely because the MGM can deal with more different kinds of data distributions because of its personalized approach. Furthermore, a lower threshold percentage seems to be more important than other parameters. However, generally threshold setting 2 does better**. Finally, a low number of features and a high (above 5) or low number of components (around 2) appears to be the best combination of parameters for the MGM.** Any combination inbetween does not do well in the experiments. The MGM does not require more than two important features, as when it captures their shapes well, the models perform better across all metrics.

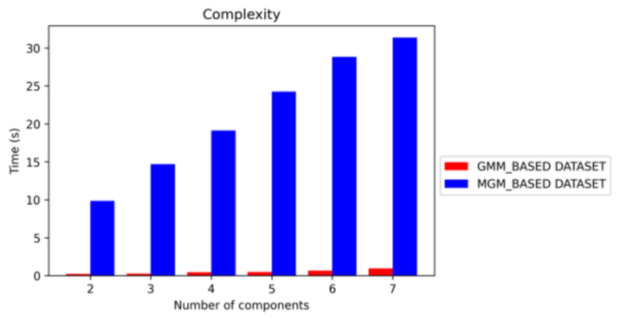
Limit cases

In many cases the MGM does not converge at all when it is fitted to 5 or more features combined with 6 or more clusters. This is likely because the MGM simply becomes too complex and subsequently, does not fit the niche dataset correctly. Finally, we also tested this with a 90%, for which the models do not converge, as extracted points do not seem to yield high enough probabilities of being found within the distribution of the niche market dataset.

Complexity Analysis

As can be seen in figure 15, the complexity of the MGM model is considerably higher than for the GMM model.

Figure 15: Complexity Analysis of the MGM and GMM model



The algorithm finds and fits the best distribution for each set of points belonging to each feature for each component. This is O(n^2) in the worst case as the iteration goes through each feature and each component of the dataset. However, as the previous results indicate, the MGM can work very well with a low number of features and components. It is therefore still a feasible alternative to a GMM based model when dealing with large datasets.

Ablation study

Removing the fitter function means we need to manually decide what functions are best suited for a set of points. This will not allow us to build an efficient MGM that is comparable in usage to a current state-of-the-art GMM.

Chapter 7

Conclusion and Future Works

We have shown that incorporating more individualized distributions into a general mixture model for data augmentation of a small dataset, can improve performance across several experimental variations. In nearly all experiments the MGM based dataset has higher precision scores than the GMM based dataset. When the conditions are exactly right, the GMM-based model does outperform the MGM model. However, the results suggest that the MGM-based models are more robust and precise with low number of features. This means that even though the MGM-based model has a higher complexity this does not affect the speed of receiving precise results as the GMM-based models generally requires a greater number of features for precise results. Specifically in settings where there is little data available, the MGM-based approach can therefore allow for high precision scores without immense computational overhead.

Future Works

Combined with a greater selection of distributions, the MGM model presented in this paper has immense potential to outperform existing state-of-the-art models. It would be highly beneficial if pomegranate were to implement support for all distributions available in the Fitter library, to allow for more optimal fitting procedures for the MGM. It is also suggested to use methods such as threading and multiprocessing to reduce the computational overhead of assigning individualized distributions to each feature. Furthermore, the results could be even more conclusive for other classifiers or regression tasks. This needs to be tested. In addition, creating a function that optimizes the combination of components and features, while computationally heavy, would allow for identifying the ideal combination of the two for a given setting. Also, various more thresholding techniques could be tested for the two approaches, and one could use the entire client dataset to form the initial MGM. Moreover, more experiments can be conducted with different variations of niches. It should also be tested whether categorical features can be fed and modelled with the MGM. Also, it would be advised to test an MGM with for example 5 gaussian components against one with 4 gaussian components and one different one to see whether the data can be modelled even more precisely. Finally, because of the importance of maintaining a balanced target column, future work could investigate adding more extracted points with target value of the underrepresented target.

Challenges

This investigation has shown that standardized techniques for returning and reusing pandas' datasets need to be formalized. Also, since the models are based on probabilities, the results change every time a new experiment is run. This makes it hard to draw conclusions from the results.

Contributions and Acknowledgements

In this investigation, we designed a novel architecture for data augmentation that automatically adds components with initialized parameters into an MGM model. We do so by assigning clusters to each row, returning the best distribution using python's fitter library in the format of pomegranate for a given set of points. We develop a method applying this process for all clusters and points in the S domain. Furthermore, we develop two thresholding approaches and reporting functions. These reporting functions allow us to choose what thresholding approaches to use, with what percentages and whether to experiment with the number of components or the number of features. We also create various plotting functions for results and overlaps between augmented datasets.

For figures in the experiments section 1, we use pomegranate's plotting function from their documentation. We also make use of a function by Berke Filiz for determining the ideal number of components for a given dataset, and by Alice Hamberger for determining the overlap between extracted entries and plausible entries assigned at the start. We also utilize Alice Hamberger's parameters for the complex niche for this experiment. Both are students in the capstone working group of MLP. Lastly, we make use of functions provided by MLP to create a hard and complex niche domain and train our LGBM.

Dataset

https://www.kaggle.com/code/fathinahizzati/credit-default-risk-1/notebook

Works Cited (APA)

Aggarwal, A., Mittal, M., & Battineni, G. (2021). Generative adversarial network: An overview of theory and applications. International Journal of Information Management Data Insights, 1(1), 100004–100004. doi:10.1016/j.jjimei.2020.100004

Arora, A., Shoeibi, N., Sati, V., González-Briones, A., Chamoso, P., & Corchado, E. (2021). Data augmentation using gaussian mixture model on csv files. Advances in Intelligent Systems and Computing. doi:10.1007/978-3-030-53036-5_28

Bengio, Y., Courville, A., & Vincent, P. (2012). Representation Learning: A Review and New Perspectives. http://arxiv.org/abs/1206.5538

Bowles, C., Chen, L., Guerrero, R., Bentley, P., Gunn, R., Hammers, A., … Rueckert, D. (2018). GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks. http://arxiv.org/abs/1810.10863

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics, 21(1). doi: 10.1186/s12864-019-6413-7

Dash, S. (2022). Brier Score - How to measure accuracy of probablistic predictions - Machine Learning Plus. Retrieved 31 May 2022, from https://www.machinelearningplus.com/statistics/brier-score/

Erlich_bachman Khandelwal. (2017). Which algorithm takes the crown: Light GBM vs XGBOOST?. https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/

Fürnkranz, J., Chan, P. K., Craw, S., Sammut, C., Uther, W., Ratnaparkhi, A., … De Raedt, L. (2011). Mixture Model. Encyclopedia of Machine Learning (680–682). doi:10.1007/978-0-387-30164-8_547

Hemmati, S., Capak, P., Pourrahmani, M., Nayyeri, H., Stern, D., Mobasher, B., … Shahidi, A. (2019). Bringing Manifold Learning and Dimensionality Reduction to SED Fitters. The Astrophysical Journal, 881(1), L14–L14. doi:10.3847/2041-8213/ab3418

Institute of Electrical and Electronics Engineers. (2019). IEEE International Conference on Systems, Man and Cybernetics (SMC) : Bari, Italy. October 6-9, 2019.

Jason Brownlee. (2019). A Gentle Introduction to Expectation-Maximization. https://machinelearningmastery.com/expectation-maximization-em-algorithm/

Jyoti Prakash Maheswari. (2019). Breaking the curse of small datasets in machine learning part 2. https://towardsdatascience.com/breaking-the-curse-of-small-data-sets-in-machine-learning-part-2-894aa45277f4

Lee, K., Zhu, Y., Sohn, K., Li, C.-L., Shin, J., & Lee, H. (2020). i-Mix: A Domain-Agnostic Strategy for Contrastive Representation Learning. http://arxiv.org/abs/2010.08887.

Malinin, A., Band, N., Ganshin, Alexander, Chesnokov, G., Gal, Y., … Yangel, B. (2021). Shifts: A Dataset of Real Distributional Shift Across Multiple Large-Scale Tasks. http://arxiv.org/abs/2107.07455.

Nguyen, T. (2011). Scholarship at UWindsor Gaussian Mixture Model based Spatial Information Concept for Gaussian Mixture Model based Spatial Information Concept for Image Segmentation Image Segmentation. https://scholar.uwindsor.ca/etd

Pan, L., Li, Y., He, K., Li, Y., & Li, Y. (2020). Generalized linear mixed models with Gaussian mixture random effects: Inference and application. Journal of Multivariate Analysis, 175. doi:10.1016/j.jmva.2019.104555

Reynolds, D. (2009). Gaussian Mixture Models. Encyclopedia of Biometrics (σσ. 659–663). doi:10.1007/978-0-387-73003-5_196

Saptashwa Bhattacharyya. (2020). Latent Variables & Expectation Maximization Algorithm. https://towardsdatascience.com/latent-variables-expectation-maximization-algorithm-fb15c4e0f32c

Schreiber, J., & Allen, P. G. (2018). pomegranate: Fast and Flexible Probabilistic Modeling in Python. https://github.com/jmschrei/pomegranate

Sklearn.mixture.BayesianGaussianMixture. (n.d.). Bayesian Mixture. https://scikit-learn.org/stable/modules/generated/sklearn.mixture.BayesianGaussianMixture.html#sklearn.mixture.BayesianGaussianMixture

Sklearn.metrics.precision_score. (2022). https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html

Zhang, Jing, et al. '3D Reconstruction for Motion Blurred Images Using Deep Learning-Based Intelligent Systems'. Computers, Materials & Continua, vol. 66, no. 2, 2021, pp. 2087–104. DOI.org (Crossref), https://doi.org/10.32604/cmc.2020.014220.

Appendix A

Overview of Libraries

Sci-kit learn is a library that allows us to perform unsupervised clustering with methods like GMMS. We will use this for assigning a cluster to entries in the dataset. To utilize the shape of a set of points of a feature in our MGM we need a library that automates the process of finding the best distribution. Pythons' *fitter library* provides access to over 50 distributions and assigns an error score of choice to each distribution. For this investigation we will use the sum-square error to select the most optimal fit for a given set of points. Finally, we use *pomegranate* to make our General Mixture model. From probability distributions to Bayesian networks and hidden Markov models, pomegranate maintains rapid and adaptable probabilistic models. Pomegranate's underlying principle is that all probabilistic models are probability distributions in the sense that they all provide probability estimates for samples and are updated with samples and weights. The key benefit of this viewpoint is that pomegranate components may be stacked more freely than those in other packages. A Gaussian mixture model, for example, is just as simple to construct as an exponential or log normal mixing model.

Appendix B

Experiments with the Complex Niche

Key: #of trials, threshold setting, number of clusters or features, threshold percentage.

Varying features
3 thresh2 3 Clust 80
3 thresh0 3 clust 80
3 thresh2 4 clust 80
3 thresh0 4 clust 80
3 thresh2 5 clust 80
3 thresh0 5 clust 80
3 thresh2 3 Clust 70
3 thresh0 3 clust 70
3 thresh2 4 clust 70
3 thresh0 4 clust 70
3 thresh2 5 clust 70
3 thresh0 5 clust 70

Varying clusters
3 thresh2 2 feat 80
3 thresh0 2 feat 80
3 thresh2 3 feat 80
3 thresh0 3 feat 80
3 thresh2 5 feat 80
3 thresh0 5 feat 80
3 thresh2 2 fest 70
3 thresh0 2 feat 70
3 thresh2 3 feat 70
3 thresh0 3 feat 70
3 thresh2 5 feat 70
3 thresh0 5 feat 70

Appendix C

Experimental results Hard Niche

Figure 16a: 80% threshold, components : 3

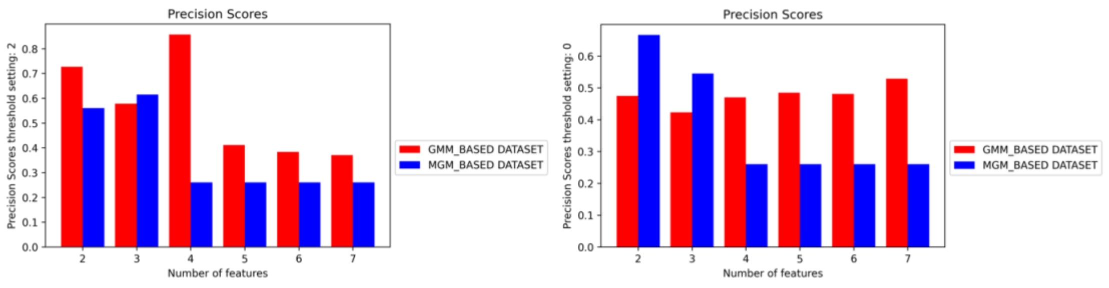


Figure 16b: 70% threshold, components : 3

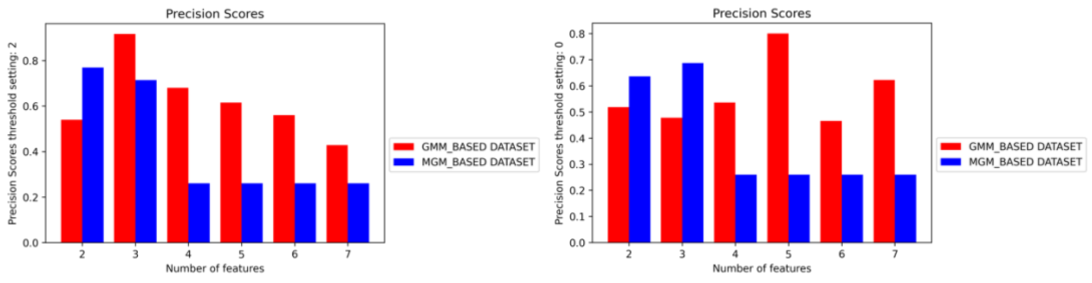
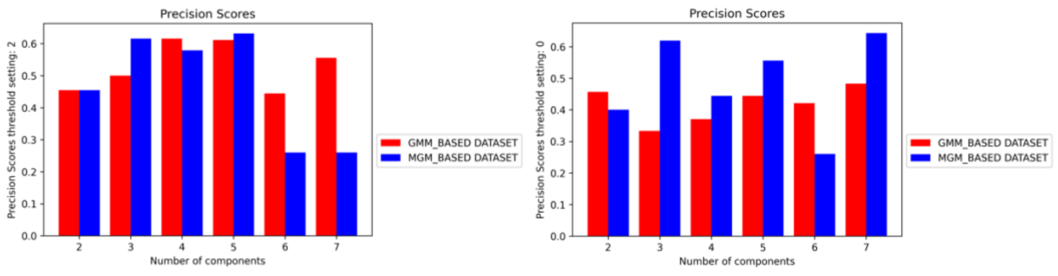


Figure 16c: 80% threshold, features : 4


Precision Scores
Precision Scores threshold setting: 2
Number of components
GMM_BASED DATASET
MGM_BASED DATASET
Precision Scores
Precision Scores threshold setting: 0
Number of components
GMM_BASED DATASET
MGM_BASED DATASET