Generating Synthetic Data using Entropic Marginal Matching

Lucas Jevtić

Wolfson College

University of Oxford

A dissertation submitted for the degree of

MSc Mathematical Modelling and Scientific Computing

1st of September 2021

Acknowledgements

First and foremost, I would like to acknowledge my supervisors Professor Terry Lyons and James Morill for all their help, advice, suggestions, and patience throughout this project. It has been an honour and pleasure to work with you both.

Thank you to my friends on the MMSC course for the lively discussions, endless banter, and the welcomed distraction of our table football games.

To Dr Kathryn Gillow for looking after all of us on the MMSC, and for making this course the best it could be in a challenging year.

To my parents for their continuous encouragement and support, for always challenging me to produce my best work, and for teaching me the value of education and hard work.

Lastly, I would like to thank Alfie, who has always been by my side when I have needed support, provided encouragement, and is my faithful and noble companion. His acute insight and inquisitive nose have led to many novel discoveries; some mathematical but most food related. Without him, none of this would have been possible. He has always believed in me, showing his faith with an endless supply of soggy socks and tennis balls.

Figure A: Alfie, the Poobrador

Abstract

Many research applications require access to personal data, however, its availability has become increasingly limited due to data protection laws; making synthetic data an attractive alternative. While many synthetic data generation methods exists, few have the ability to produce feasible samples without access to the original data. By extending existing survey-weighting-techniques, this work proposes a method for generating realistic synthetic data using class-conditional summary statistics. To create the synthetic data, weights are allocated to samples of a large dataset that is representative of some general population, referred to as the *corpus dataset*. The weights are chosen such that the statistics of the synthetic data match target summary statistics, while maintaining a desirable distribution of the weights. The method is formulated as a convex optimisation problem, balancing two objectives: matching of target statistics and regulating the entropy of the weights' distribution. Hence, the method has been called *Entropic Marginal Matching (EMM)*. We demonstrate the application of this new methodology by generating synthetic data for diabetes and COVID-19 datasets. These initial experiments provide promising evidence that EMM can produce synthetic data that has a high statistical resemblance to the target dataset it is attempting to replicate.

Contents

Chapter 1

Introduction

Data has become one of the most valuable commodities of our time. Every day, the world creates 2.5×10^{18} bytes (2500 petabytes) of data. The amount of data produced each year has been rapidly increasing, with 90% of all data that exists having been created in just the last two years [7]. With this growth of data volume, there has been a rapid advancement of the science, technologies, algorithms and ethics behind big data. In particular, the ethical discussion of the rights of an individual with respect to their personal information has led to significant changes in governmental laws pertaining to data privacy. Legislatures such as GDPR in the European Union [9], PIPEDA in Canada [36], and the UK's Data Protection Act [37] all impose a weighty legal responsibility on those who work with data under those governments' jurisdiction.

Some of the most interesting tasks in machine learning necessitate the use of an individual's medical, behavioural, or socio-demographic information. For example, it is unreasonable to expect to be able to predict adverse health events without the collection of some form of personal data, be that demographic information like age, or medical information such as heart rate [46]. However, such data poses significant risks for re-identification; Rocher et al. [43] found that 99.98% of Americans could be correctly re-identified in any dataset with 15 or more demographic attributes. As such, extra care is required to follow data privacy laws to mitigate the risks associated with re-identification [15].

A mounting concern is the inaccessibility of useful datasets due to the requirement that they abide by these stringent data privacy requirements. Even when such data may be accessed, the assurance of correct data usage and protection is frequently a protracted process with firm legal requirements. Often, this can slow the momentum

of time-sensitive research to the point that it may significantly restrict the effectiveness of its outcome [11].

The desire for sensitive patient data to be accessible to others has resulted in increased interest in alternatives to accessing the raw form of the real data. The primary aim is to provide the data in a format in which the data is anonymised; that is, the data cannot be used to identify any individual in the dataset. Methods for doing this include removing identifiable features, adding noise to potentially identifiable variables, or creating broader categories of the features [52]. However, proving that any resulting expression of the original data is non-identifiable is challenging. In [8], the authors show that many datasets still exhibit a high identification rate even when attempts have been made to anonymise them.

An alternative method is to use synthetic data generation techniques which represent a promising addition to the more straightforward anonymisation techniques mentioned above. The basic aim is to use some 'data privacy preserving' set of information about the real-data, such as a set of class-conditional summary statistics, and use this to generate a synthetic set of data that contains these properties. This data can then be shared and used to train learning algorithms without the risk of sharing personally identifiable information.

The focus of this work is towards the improvement of synthetic data generation techniques for problems in health care, which currently lags behind other industries in information technology, data exchange, and interoperability [54]. In this sector, data-driven research depends on access to high-quality health datasets, however, as mentioned above, access to real electronic health records is subject to stringent data protection laws. Re-identification, even from anonymised health records, is a significant risk factor and has been observed on multiple occasions [8, 34, 48]. The response to events such as these has reduced the number of datasets, anonymised or otherwise, that are openly available for research and development purposes.

In a nutshell, the work in this document defines a new synthetic data generation method for medical data that performs a re-weighting of freely available datasets to align with summary information about the private dataset of interest. This results in a collection of samples that not only mimic the statistics of the dataset of interest, but also contains only realistic patient examples as they are drawn from a real collection of patients, a feature not found in existing approaches [11]. The prior availability of these datasets alleviates the ethical complications attributed to research using

personal data while maintaining the realistic properties of individual samples. For example, the University of Oxford states:

> "If the data has been previously collected, and cannot be traced back to identifiable individuals, you need not obtain ethical approval ... Projects using existing data from a census, for example, do not need approval [42]."

Through our research of related works, we have found the methodologies surrounding survey re-weighting to be particularly applicable to this task. These involve the allocation of weights to samples in order to change the distribution of the data in to a form that is more desirable. Whilst usually applied to correct deviations of sample distributions to match those known in the population, we re-weight a large dataset representing the population, referred to as the corpus, to match conditional distributions. By adapting the recent and relatively unknown work of Barrat et al. [1], a convex optimisation problem may be constructed that finds the optimum weighting of the corpus to match desired statistics while maintaining regularity, through entropy maximisation, in the distribution of the weights. We have named this method as *Entropic Marginal Matching (EMM)*.

Chapter 2

Motivation

In this paper, synthetic data is considered to be fabricated patient records that were not obtained by direct measurement [35]. Its aim is to mimic the statistical properties of the original data source, whilst being fully anonymised in the sense that individual patients from the original data cannot be identified from it. Representation of the original data source paired with non-identifiability make it a viable substitute in various applications, particularly those in the field of machine learning [11, 57].

The use of synthetic data has several advantages, particularly when working with data sources where the safeguarding of personal information is of great importance [32]. These include:

- **Ease of access** - synthetic data may be generated on-the-fly and tailored to meet the specific needs of the application.
- **Cost-effective** - often the cost of generating (or altering) data is far less than collecting new real-world data. In applications that require a large number of data samples, it can often be impractical or impossible to collect the amount of data that is needed.
- **Maintains privacy** - By generating data rather than using real-world samples, privacy can be maintained since the data does not relate to individuals from the original data and thus cannot be used to identity them.

A recent review of current data generation methods [11, 45] shows that much work has been aimed towards developing synthetic data generation approaches for medical applications. These techniques use advanced machine learning or statistical methods, such as generative adversarial networks (GANs) or Bayesian networks (BNs) to

anonymise sensitive data. However, many of the methods (such as a GAN) require full access to the original dataset to generate a synthetic one.

Our particular motivation is the generation of synthetic data in situations when the original dataset is not available. Instead, we only have access to a collection of summary statistics of said data. Frequently, in lieu of releasing their original data, researchers will publish summary statistics tables which give an overview of the statistical properties of the data of interest without exposing any individual information, making re-identification impossible.

Table 2.1 displays an example of the type of summary statistics of interest. These statistics correspond to a range of variables conditional on whether or not the patients have cardiovascular disease, whilst the raw form of the data is inaccessible, due to data protection requirements. Our aim is to develop data generation techniques that can leverage information in the form presented in Table 2.1 to build models that can predict a variable of interest; in this case, cardiovascular disease.

The COVID-19 pandemic represents another good example of the usefulness of developing such approaches. In the early stages of the pandemic, little was known about the symptoms that presented in COVID-19 cases, and there was minimal freely available data. However, there existed many studies that published statistical tables conditional on a positive COVID-19 test analogous to Table 2.1; see [13]. Synthetic data generation that can be applied to tabular data of this form can be used to generate algorithms to provide risk and symptom information to individuals both in hospital and at-home.

One of the challenges in generating realistic synthetic data from summary statistics without access to the original dataset is the lack of information about the shape or type of distribution of each variable. Furthermore, even less information is known about the latent relationships between variables. For example, in [47] the authors generate a synthetic dataset from a summary table in [13] to predict severe vs nonsevere presentations of COVID-19. The authors use a simple Bayesian inference technique that treats all features independently. By virtue of the curse of dimensionality, this makes it highly likely that generated samples do not represent realistic patients.

For a clear example, note that systolic blood pressure must necessarily be higher than diastolic blood pressure. In addition to this hard constraint, there are also significant correlations between the two variables that are not expressed in Table 2.1. Using a naïve, feature-independent, Bayesian inference like that mentioned above does not

	Without previous cardiovascular disease (n=186 988)	With previous cardiovascular disease (n=157 728)
Sex		
Women	91 071 (48·7%)	51 905 (32·9%)
Men	95 907 (51·3%)	105 823 (67·1%)
Age (years)	65·3 (9·6)	65·7 (8·9)
Systolic blood pressure (mm Hg)	157 (21)	146 (20)
Diastolic blood pressure (mm Hg)	89 (12)	84 (11)
Categories of systolic blood pressure (mm Hg)		
<120	4794 (2·6%)	11 839 (7·5%)
120–129	10 134 (5·4%)	19 400 (12·3%)
130–139	18 935 (10·1%)	27 524 (17·5%)
140–149	31 342 (16·8%)	32 216 (20·4%)
150–159	31 843 (17·0%)	26 768 (17·0%)
160–169	38 829 (20·8%)	20 180 (12·8%)
≥170	50 941 (27·3%)	19 677 (12·5%)
Categories of diastolic blood pressure (mm Hg)		
<70	9614 (5·1%)	13 804 (8·8%)
70–79	26 900 (14·4%)	35 288 (22·4%)
80–89	52 663 (28·2%)	55 004 (34·9%)
90–99	54 177 (29·0%)	37 473 (23·8%)
100–109	33 514 (17·9%)	13 182 (8·4%)
≥110	9936 (5·3%)	2849 (1·8%)

Table 2.1: An example of the summary statistic tables often found in medical papers. Source: [40].

enforce this constraint and will likely result in a number of unrealistic cases. When generating blood pressure values from normal distributions defined with the means and standard deviation given in Table 2.1, approximately 0.33% of samples violated the hard constraint. In the case where this constraint is known, mitigation of invalid samples is achieved easily; simply discard all samples where the constraint is violated. However, in cases involving more complicated datasets with multiple inter-variable relationships that are not known apriory, it becomes highly taxing, if not impossible, to verify that these constraints are satisfied [28].

The property of realism regarding synthetic data is seen to provide a greater level of reliability, accuracy, and validity in its application [3, 55]. However, out of 7,746 synthetic data generation articles, only 296 were found to contain the terms realism or

realistic. Even when additional terms which were similar in meaning where searched for, such as "authentic" or "accurate to real structures", approximately only one-third of all synthetic data generation articles contained these terms [28]. Our aim is to construct a framework for a synthetic data generation method focused on preserving data realism, and in which all values produced are valid and where synthetic samples are indistinguishable from real samples. Importantly, we also wish to achieve this with only access to the summary statistics of the original data, and not the original data itself.

As we have already alluded, our generation method relies on the re-weighting of an existing corpus of patient data. We use the word corpus to refer to a large sample of the general population that contains feature overlap with the real data that our synthetic dataset will attempt to mimic.

By performing a re-weighting of real data, we ensure that every sample under consideration represents a realistic case for what a patient can feasibly look like. Assuming the corpus to be of a high data quality, we will never be considering a patient that has a set of characteristics that could not be observed in the real-world. For example, we will never have patients that have their systolic blood pressure lower than their diastolic blood pressure. Additionally, the data is likely to exhibit the latent correlations that exist between variables.

With the wealth of data publicly available, we expect many corpora to exist, however, finding those with all the required features does present a significant challenge. From our research, we have found the following for general demographic uses: the USA census: Public Use Micro Sample (PUMS) [51], the Adult corpus [23], the Historie de vie survey [18], and the Midlife in the US survey [26].

For applications in the health sciences, we have found the Hospital admission dataset published in [17] to be very complete. It contains 560,486 instances and 972 variables regarding demographics, triage evaluation, chief complaints, and past medical history. Other datasets, such as the Medical Information Mart for Intensive Care (MIMIC) are freely available, comprising of anonymised health-related data from over 40,000 patients. The data includes vital signs, medication, laboratory measurements, observations and notes by care providers, and much more [20].

Chapter 3

Related work

As introduced Chapters 1 & 2, the ability to produce synthetic data has become a valuable tool for use academics (and other agents) to gain insights and construct models. In this chapter, an overview is provided of the main methods for synthetic data generation; drawing from our own research and the recently published (2020) literature reviews of Gonclaves et al. [11], and Yale et al. [57].

Creation of synthetic data can be separated into the following two classes:

- **Process-driven methods** – Mathematical models or estimated distributions are used to generate new data. These methods include sampling from independent marginals, numerical simulations, Monte Carlo simulations, and agent-based modelling [11].
- **Data-driven methods** – These generate synthetic data through the altering of observed data. These methods encompass those using machine learning models such as Generative Adversarial Networks (GANs) or Bayesian networks (BNs), as well as the adapted methods we explore from survey methodology.

3.1 Modelling methods

If access to the full dataset is available, a common approach is to fit a distribution to the data from which samples can be drawn. Various methods exist to fit distributions to the datapoints. For example, a simple strategy is to fit some distribution to each of the features independently; however, this will often lead to unrealistic samples being drawn as it is difficult to capture the inter-attribute relationships and correlations

that are required to keep the samples grounded in reality. This becomes increasingly problematic the larger the dimension of the data [57].

A straightforward approach is to assume that the data can be modelled using a multivariate Gaussian distribution [57]. While simplistic, it achieves the rudimentary goal of providing an approach to draw samples that share certain characteristics with the real data. The shared characteristics being the means and variances – or better still the co-variances – for each variable/between variables. This method maintains a high level of privacy as it has only accessed broad summary statistics (mean/variance/co-variances) about the data, and no actual data samples.

Generation by drawing from probability distributions may be improved through the use of rule-based methods [56]. The data is still sampled from distributions, but specific rules are encoded to ensure only samples that follow a user-defined set of constraints are accepted. While rule-based methods have the ability to generate more feasible data, this comes at a cost as the rules must be created manually. Thus, it is difficult and time consuming to use these methods when working with a great number of variables. Moreover, it relies on an experienced user to correctly identify the many rules needed to produce realistic data.

The major drawback of such approaches is enforcing realism of the generated samples. For large numbers of inter-related features, estimating a realistic distribution is challenging – we must have precise information on the degree of inter-relatedness between each feature, else we are likely to produce samples in unrealistic regions of the feature space. Reconsider the blood pressure example from Chapter 2, even if we enforce that systolic is greater than diastolic blood pressure to remove impossible cases, we still have not adequately captured the correlation strength between the two features. Moreover, we additionally need to encode their relationship to other features, and higher order relationships that vary dependent on the difference between the blood pressures.

3.2 Weighting methods

In survey methodology, researchers frequently employ a number of statistical weighting techniques to adjust samples so that they better represent the population [29]. For example, in opt-in surveys, weighting methods are frequently used to rebalance the results so that potential response biases can be removed, such as a greater female response rate. These methods are advantageous as they do not rely on creating new

samples to change the distribution of the data. This ensures that all the data remains realistic.

Weighting methods within survey methodology primarily focus on matching row and column sums of cross-tabulations, see Example 3.2.2. Thus, some adaptation is needed for these methods to be applied to summary statistics of the form given in Table 2.1. Furthermore, to the best of our knowledge, these methods have not been used to the medical domain, nor have they been extended to incorporate a wider range of summary statistics, such as higher order moments.

Below, the mathematical definitions of a dataset and a weighted dataset have been given, as well as an example of how survey weighting can be used to correct male to response ratios.

Definition 3.2.1 (Dataset)**.** If $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ where each instance $\mathbf{x}_i \in \mathbb{R}^d$, then we call $\mathcal{X}$ a dataset of length N.

Definition 3.2.2 (Weighted dataset)**.** Let $\mathcal{X}$ be a dataset of length N. Additionally let $\mathbf{w} = (w_1, \ldots, w_N)$ where $w_i \in \mathbb{R}_{\geq 0}$ be such that w_i is associated to $\mathbf{x}_i$. Then we call $\mathbf{w}$ a collection of weights and we call the collection $(\mathcal{X}, \mathbf{w})$ a weighted dataset.

Example 3.2.1 (Survey weighting)**.** Suppose a survey $\mathcal{X}$ has 10 respondents denoted $\mathbf{x}_i$. There are 4 males and 6 females, hence the proportion of males to females is 0.4. However, we know that the population proportion is 0.5, therefore by allocating all female respondents a weight of 5/6, and all male respondents a weight of 5/4, we effectively change the proportion of males versus females to 0.5. Alternatively, we can express the number of male and female respondents in the weighted survey $(\mathcal{X}, \mathbf{w})$ using an indicator function,

$$\begin{aligned} \text{No. of males} &= \sum_{i=1}^{10} w_i \, \mathbf{1}_{\text{Male}}(\mathbf{x}_i) & \text{No. of females} &= \sum_{i=1}^{10} w_i \, \mathbf{1}_{\text{Female}}(\mathbf{x}_i) \\ &= \frac{5}{4} \times 4 = 5, & &= \frac{5}{4} \times 4 = 5. \end{aligned}$$

3.2.1 Iterative proportional fitting

The most common technique in survey weighting is *Iterative Proportional Fitting (IPF)*, which is also known as *raking* [29]. Originally demonstrated by Demming and Stephen [5] as a method of re-weighting a dataset $\mathcal{X}$, such that $(\mathcal{X}, \mathbf{w})$ is as similar to $\mathcal{X}$ as possible, while also having some target row and column totals.

It works by matching the marginal population totals of desired features while also minimising the relative entropy between $\mathcal{X}$ and $(\mathcal{X}, \mathbf{w})$. IPF achieves this by cycling over each feature and performing post-stratification to iteratively update the weights until the target marginal population totals are matched.

Example 3.2.2 (Matching marginal totals)**.** Suppose a survey consists of 100 responses, two of the attributes recorded are sex and home location. In the top left-hand corner of Table 3.1 the number of respondents for each of the variable combinations are shown. The survey is re-weighted to match target totals of 49:51 of males to females, and 57:43 of those living in rural areas versus urban areas. Each respondent is given a sample weight corresponding to which marginal they belong to. For example, every male who lives in an urban area is given a weight of 0.6, and every male who lives in a rural area a weight of 1.3. Hence, the total number of males in the weight survey is $20 \times 1.3 + 39 \times 0.6 \approx 26 + 23 = 49$, equal to the male target total.

	Male (weights)	**Female** (weights)	**Survey totals**	**Weighted totals**	**Target totals**
Rural	20 (1.3)	17 (1.8)	37	57	57
Urban	39 (0.6)	24 (0.8)	63	43	43
Survey totals	59	41			
Weighted totals	49	51			
Target totals	49	51			

Table 3.1: Cross-tabulations of survey respondents, where survey totals have been re-weighted to match the target totals.

3.2.2 Representative Sample Weighting

Barrat et al., in their recent (2021) and unknown paper (zero citations) [1], formulate a general approach for choosing weights that balances two objectives: matching some desired statistics to make the weighted sample representative of the population, and having weights with desirable properties. They construct their method for choosing weights as an optimisation problem. This allows for greater flexibility in the choice of measure for how representative a weighted dataset is, and for how one wishes the weights to be distributed.

The flexibility and interpretability of their approach makes it an appealing foundation for our application. It provides an adaptable method to re-weight corpora to match class-conditional statistics such as those presented in the previous chapter. Thus, we have chosen not to outline the mathematics of Representative Sample Weighting here, and instead incorporate it into the mathematical formulation of EMM in Chapter 4.

3.3 Machine learning methods

The literature around synthetic data generation is rich in its description of methods which rely on the availability of the original dataset to train a data generator to learn the patterns of the original dataset [57]. While these techniques are state of the art, they have a distinct lack of applicability to our problem as they require full access to the original dataset. We give a brief discussion here to give a fuller view of the range of techniques available.

Bayesian Networks

Bayesian networks are probabilistic graphical models, composed of acylic graphs, where the nodes represent random variables, and the edges encode the existence of probabilistic causal dependencies between the nodes [38]. The graph's structure is learnt through training on the original data, and can provide a visual interpretation of inter-variable relationships. Synthetic data is generated by sampling from the inferred Bayesian network [11].

Variation Autoencoders (VAEs)

The VAE, introduced by Kingma and Welling in [22], is a type of deep neural network used for learning the latent distributions in data. VAEs are a generative model that learn to represent data using Guassian distributions by transforming the data into a lower dimensional space, and then transforming the data from the low dimensional space back to the dimension of the original input [44]. Under the limitations on the information imposed by the lower dimensional space, VAEs learn to summarise the important characteristics of the input data instead of memorising samples used in training. Since the low dimensional layer is stochastic as it is based on Gaussian distributions, the model is generative.

Generative Adversarial Networks (GANs)

GANs, introduced by Goodfellow et al. in [12], have produced successful results when generating synthetic data from complex data types such as images or text [58, 41]. They work by training two neural networks: a generative network G and a discriminator network D. The generative network takes an input of a random variable, often drawn from a simple distribution, and outputs data of the same dimension as the original data. The discriminator network attempts to distinguish between the generated data and the original data [12].

Chapter 4

Mathematical formulation

4.1 Entropic Marginal Matching

Extending the techniques outlined in [1], we have constructed a framework for data generation, orientated towards use in machine learning applications. Specifically, we have designed it for cases where we have summary statistic tables in the form of Tables 2.1 6.4 and 6.7. These summary statistics are presented on a range of features conditional on class labels.

We formulate this problem by supposing there exists a private dataset $\widehat{\mathcal{X}}$, with class labels k for $k \in \{1, \ldots, M\}$ for which we only have access to some collection of summary statistics $\mathcal{F} = \{\hat{\mathbf{f}}_1, \ldots, \hat{\mathbf{f}}_M\}$ (similar to Table 2.1). We refer to $\widehat{\mathcal{X}}$ as the target dataset, and $\hat{\mathbf{f}}_k$ as the target statistics for class k. We assume that there also exists a publicly available corpus dataset $\mathcal{X}$ with N instances, such that $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ with $\mathbf{x}_i \in \mathbb{R}^d$. For example, the dataset that produced the summary statistics in Table 2.1 can be seen as the private dataset $\widehat{\mathcal{X}}$, and the hospital admissions dataset as the public corpus $\mathcal{X}$.

The primary aim of EMM is to generate, for each class k, a weighted corpus from $\mathcal{X}$ such that the weighted summary statistics align with the target statistics $\hat{\mathbf{f}}_k$, made available to us from $\widehat{\mathcal{X}}$. As in Section 3.2, a weighted corpus is defined as $(\mathcal{X}, \mathbf{w})$, where $\mathbf{w} = (w_1, \ldots, w_N)$ with $w_i \in \mathbb{R}_{\geq 0}$ such $\sum_{i=1}^{N} w_i = 1$. By enforcing $\sum_{i=1}^{N} w_i = 1$ we allow the weights to be used to denote the probability of observing a given sample, that is, $\mathbb{P}(\mathbf{x}_i) = w_i$.

Once the corpus have been re-weighted to match the target statistics $\hat{\mathbf{f}}_k$, all instances are given class label equal to k. This method is then repeated for all labels, to create

M weighted corpora. Finally, the synthetic dataset is created by concatenating all weighted corpora to form a dataset with $M \times N$ instances.

We can easily create a synthetic dataset without weights by sampling individuals from the weighted synthetic dataset with probability $w_i / \sum_{i=1}^{M \times N} w_i$. Alternatively, many machine learning models can accept sample weights which modify the contribution of a given sample to the overall loss; as such, machine learning models can be built directly from the weighted synthetic dataset without the need for sampling.

In general, this is an under-determined problem; that is, there are multiple choices of weights that result in matched summary statistics. However, many choices for weights result in feature distributions of the weighted corpus that are un-realistic. For this reason, an adjustable penalty function is used to maximise the entropy of the weights while matching target statistics. Hence, we refer to our method as *Entropic Marginal Matching*.

4.2 Weights and expectations

The rational behind re-weighting data is that it changes the representation of individual samples, allowing the original distribution of a dataset to be augmented into another. We may think of an unweighted dataset as a dataset where all samples have equal weighting and thus all have equal probability of selection. By changing the weights, we are able to alter the probability of selection and thus the distribution of the data.

Example 4.2.1 (Weighted expectation)**.** Given a weighted corpus we define the expectation of the corpus to be

$$\mathbb{E}[(\mathcal{X}, \mathbf{w})] = \sum_{i=1}^{N} w_i \mathbf{x}_i. \tag{4.1}$$

In the case where we have equal probability weights, that is $\mathbf{w} = (\frac{1}{N}, \ldots, \frac{1}{N})$, this reduces onto the expectation of the unweighted corpus.

We generalise the idea of the weighted expectation onto the idea of a weighted summary statistic of the corpus.

Definition 4.2.1 (Weighted summary statistics)**.** If $f : (\mathcal{X}, \mathbf{w}) \to \mathbb{R}$ such that

$$f = \sum_{i=1}^{N} w_i F(\mathbf{x}_i) \tag{4.2}$$

where $F : \mathcal{X} \to \mathbb{R}$, then we call f a weighted summary statistic of the corpus $\mathcal{X}$.

In the definition above we enforce that $F : \mathcal{X} \to \mathbb{R}$, rather than allowing $F : (\mathcal{X}, \mathbf{w}) \to \mathbb{R}$. While this removes our ability to express all summary statistics, it preserves disciplined convex programming (DCP) rules required when formulating our problem as a convex optimisation in Section 4.3.

Definition 4.2.2 (Weighted n$^{\text{th}}$ order moments)**.** Note first that if we take $F(\mathbf{x}_i) = \mathbf{x}_i$, then $\mu = f = \mathbb{E}[\mathbf{x}]$. The n^{th} order moment may be calculated using $F(\mathbf{x}_i) = (\mathbf{x}_i - \mu)^n$. Hence, the central moments of the j^{th} feature may be expressed as

$$f = [\tilde{\mu}^j]^n = \sum_{i=1}^{N} w_i (\mathbf{x}_i^j - \mu^j)^n, \tag{4.3}$$

where

$$\mu^j = \sum_{i=1}^{N} w_i \mathbf{x}_i^j. \tag{4.4}$$

However, we cannot directly express weighted higher order moments under the constraints set by Definition 4.2.1, namely, that in the weighted sum must be expressed as $\sum_i w_i F(\mathbf{x}_i)$, not $\sum_i w_i F(\mathbf{x}_i, \mathbf{w})$. We propose the use of approximated weighted moments by substituting μ^j with the target mean $\hat{\mu}^j$, computing the central moments, and then standardising if necessary.

Example 4.2.2 (Approximated weighted moments)**.** Suppose the constant $\hat{\mu}^j$ is the target mean for the j^{th} feature. To compute the approximate weighed variance (2nd order central moment), we may construct f as

$$f = \sum_{i=1}^{N} w_i (\mathbf{x}_i^j - \hat{\mu}^j)^2. \tag{4.5}$$

This provides a reasonable approximation if $\mu^j \approx \hat{\mu}^j$. Since $\hat{\mu}^j$ is a constant, this preserves the DCP rules that are required for the problem to be convex.

The probability of occurrence of event A in $\mathcal{X}$ can be calculated by letting F be the indicator function, such that

$$F(\mathbf{x}_i) = \mathbf{1}_A(\mathbf{x}_i) := \begin{cases} 1 & \text{if } \mathbf{x}_i \in A, \\ 0 & \text{if } \mathbf{x}_i \notin A. \end{cases} . \tag{4.6}$$

.

Example 4.2.3 (Weighted event probability)**.** The indicator function may used to represent the event that a data point falls within a bin. Suppose we have a corpus $\mathcal{X}$, where instances of the j^{th} feature, $\mathbf{x}_i^j \in \mathbb{R}$, range from 0 to 100. To create a bin of $0 \leq \mathbf{x}_i^j < 20$, we define $F(\mathbf{x})$ as

$$F(\mathbf{x}_i) = \begin{cases} 1 & \text{if } 0 \leq \mathbf{x}_i^j < 20, \\ 0 & \text{otherwise.} \end{cases}. \tag{4.7}$$

With this definition of $F(\mathbf{x}_i)$, the may obtain the probability that $\mathbf{x}_i^j$ falls with in the bin using the Equation (4.2).

The definition of f, from Equation (4.2), maybe rewritten in the computationally convenient vector form. We define the vector

$$\mathbf{F} = (F(\mathbf{x}_1), \ldots, F(\mathbf{x}_N))^{\mathrm{T}} \tag{4.8}$$

and therefore we may write $f = \mathbf{F}^{\mathrm{T}}\,\mathbf{w}$, where $\mathbf{w} = (w_1, \ldots, w_N)^{\mathrm{T}}$. Furthermore, when computing a number of summary statistics, given by $F_1, \ldots, F_k$, then it is often useful to define $\mathbf{F}$ as the matrix

$$\mathbf{F} = \begin{pmatrix} F_1(\mathbf{x}_1) & \cdots & F_k(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ F_1(\mathbf{x}_N) & \cdots & F_k(\mathbf{x}_N) \end{pmatrix} \tag{4.9}$$

Thus, we may also define the vector $\mathbf{f} = \mathbf{F}^{\mathrm{T}}\,\mathbf{w}$.

Example 4.2.4 (The Matrix form of $\mathbf{F}$)**.** To express the weighted mean μ^j and approximate weighted variance $\widetilde{\text{var}}^j$ for the j^{th} of a weighted corpus $(\mathcal{X}, \mathbf{w})$, the matrix $\mathbf{F} \in \mathbb{R}^{N\times 2}$ is written as

$$\mathbf{F} = \begin{pmatrix} \mathbf{x}_1^j & (\mathbf{x}_1^j - \hat{\mu}^j)^2 \\ \vdots & \vdots \\ \mathbf{x}_N^j & (\mathbf{x}_N^j - \hat{\mu}^j)^2 \end{pmatrix} \tag{4.10}$$

where $\hat{\mu}^j$ is the target mean for the j^{th} feature. Hence, we have that

$$\begin{pmatrix} \mu^j \\ \widetilde{\text{var}}^j \end{pmatrix} = \mathbf{F}^{\mathrm{T}}\mathbf{w} \tag{4.11}$$

4.3 Optimisation

We formulate our task as an optimisation problem, integrating two objectives: matching of marginals and regulating the distribution of weights.

$$\begin{aligned} \min \quad & \ell(\mathbf{f}, \hat{\mathbf{f}}) + \lambda\, r(\mathbf{w}) \\ \text{s.t.} \quad & \mathbf{f} = \mathbf{F}^{\mathrm{T}}\, \mathbf{w}, \\ & \sum_{i=1}^{N} w_i = 1, \quad w_i \geq 0 \end{aligned} \tag{4.12}$$

The objective function is comprised of two functions: the loss function and the regulariser. Each enforces a penalty for not achieving a desired goal. The loss function $\ell(\,\cdot\,,\,\cdot\,)$ is used to control the type and amount of penalty imposed based on the difference between the target statistic and those given by the re-weighted corpus. The regulariser function $r(\,\cdot\,)$ adds an additional penalty for weights that are not distributed in a desirable way. The variable λ acts as a hyperparameter that scale the amount of penalty added by the regulariser.

Additionally, we enforce two constraints upon the weights; they must be non-negative, and the sum of weights must equal 1. These maintain that the weights form a probability distribution such that the probability of selecting a sample is equal to the weight corresponding to said sample.

4.3.1 Loss functions

For our purposes, we refer to loss functions as those which pertain to marginal and statistical matching. They measure how unrepresentative, or in other words, how far the distribution of the weighted corpus is from the target statistics.

Equality loss

Perhaps the most natural loss function to define is equality loss, which simply enforces the requirement that the statistics of the weighted corpus match the target statistics exactly. The equality loss function may be written as

$$\ell(\mathbf{f}, \hat{\mathbf{f}}) = \begin{cases} 0 & \text{if } \mathbf{f} = \hat{\mathbf{f}} \\ +\infty & \text{otherwise} \end{cases} \tag{4.13}$$

It is worth noting that alternatively this loss function can be enforced as a constraint, which may be more favourable depending on the optimisation methods being used.

Inequality loss

Defined similarly to equality loss, inequality loss forces the statistics of the weighted corpus to lie within a given range of the target statistics. Inequality loss is defined as

$$\ell(\mathbf{f}, \hat{\mathbf{f}}) = \begin{cases} 0 & \text{if } \mathbf{a} \leq \hat{\mathbf{f}} - \mathbf{f} \leq \mathbf{b} \\ +\infty & \text{otherwise} \end{cases} \tag{4.14}$$

where $\mathbf{a}$ and $\mathbf{b}$ are the vectors of upper and lower limits specified by the user. In most cases, $\mathbf{a} = -\mathbf{b}$, but asymmetric cases are also possible.

Squared loss

One of most common loss functions, squared loss may easily be applied here. It allows for less stringent constraints on the statistics of the weighted corpus. This is particularly useful in cases where equality cannot be achieved. Squared loss if defined as

$$\ell(\mathbf{f}, \hat{\mathbf{f}}) = \|\mathbf{f} - \hat{\mathbf{f}}\|^2. \tag{4.15}$$

When working with squared loss, it is important to consider standardising the data to avoid the magnitude of the loss being determined by the magnitude of the data.

Relative entropy loss

Relative entropy, also referred to as Kullback-Leibler (KL) divergence [24], is a measure of how different one probability distribution is from another. For two discrete probability distributions P and Q, defined on the same probability space Ω, the relative entropy from Q to P is defined to be

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \Omega} P(x) \log \frac{P(x)}{Q(x)}. \tag{4.16}$$

As a result of Gibbs' inequality, $D_{\mathrm{KL}} \geq 0$, with equality if and only if $P = Q$ almost everywhere [27].

We propose two approaches to the use of relative entropy as a loss function. One being focused on cases where probabilities are provided for categorical or binned data. The other may be used to match means and standard deviations when dealing with continuous distributions.

For categorical or binned data, if $\mathbf{f} = (f_1, \ldots, f_m)$ forms a distribution on some variable in the corpus, and hence for all $w_i \in \mathbf{w}$ we have $\mathbf{F}^{\mathrm{T}}\,\mathbf{w} \geq 0$ and that $\mathbf{1}^{\mathrm{T}}\,\mathbf{F}^{\mathrm{T}}\,\mathbf{w} =$

1 then the relative entropy between the weighted corpus and the target distribution $\hat{\mathbf{f}} = (\hat{f}_1, \ldots, \hat{f}_m)$ may be used as a loss function and is defined as

$$\ell(\mathbf{f}, \hat{\mathbf{f}}) = \sum_{i=1}^{m} f_i \log(f_i / \hat{f}_i). \tag{4.17}$$

Example 4.3.1 (Weighted bin proportions)**.** Suppose the j^{th} feature is a categorical variable with three categories: red, green, and blue. The target statistics state that 20% of instances are red, 20% are blue, and 40% are green, hence, $\hat{\mathbf{f}} = (0.2, 0.2, 0.4)$. The bin proportions of the weighted corpus are defined as $\mathbf{f} = (f_{\text{red}}, f_{\text{green}}, f_{\text{blue}})$ where f_{colour} is defined as

$$f_{\text{colour}} = \sum_{i=1}^{N} w_i \, \mathbf{1}_{\text{colour}}(\mathbf{x}_i^j)$$

In the continuous case, when aiming to re-weight the corpus distribution to match target means or standard deviations, the relative entropy between the unweighted corpus and the weighted corpus may be minimised. This results in the weighted corpus having the target statistics while maintaining a similarity to the type of distribution of the unweighted corpus. We refer to this as the Corpus KL loss function since if minimises the relative entropy between the shifted corpus and the weighted corpus.

Let the mean and standard deviation of the target feature in the unweighted corpus be $\mu_C = \frac{1}{N}\sum_{i=1}^{N} x_i$ and $\sigma_C = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_C)^2}$. The target mean and standard deviation are denoted $\hat{\mu}$ and $\hat{\sigma}$. Then, we wish to minimise the relative entropy between the probability density function (PDF) of the weighted corpus, denoted $P_C(x)$ and the unweighted corpus. We shift the unweighted corpus with respect to the target mean and standard deviation, denoted $P_W(z)$ where $z = (x - \mu_C)\frac{\hat{\sigma}}{\sigma_C} + \hat{\mu}$. Expressed using the continuous definition of relative entropy as

$$D_{KL}(P_W(x) \parallel P_C(z)) = \sum_{i=1}^{N} P_W(z_i) \log \frac{P_W(x_i)}{P_C(z_i)} \, \mathrm{d}x. \tag{4.18}$$

Note that we have assumed that the distribution is symmetric or near symmetric. We also note then that the target distribution is the mean shifted distribution observed in the corpus. Since the PDF is not explicitly known for the corpus, we approximate the PDF on discrete intervals, i.e by computing the histogram. The histogram bin probability can then be expressed in the same way as Example 4.3.1.

4.3.2 Regularisers

The distribution of the weights is also important, often it is desirable for the weights to be similar in magnitude to each other, or to have high entropy. This avoids possible scenarios where a few instances are assigned very large weights while the rest are all extremely small. Having weights that are all of a similar magnitude prevents a few samples from dominating the behaviour of the weighted distribution. There are two regularisers that may be used to introduce a penalty for nonevenly distributed weights: an upper and lower bound on the size of the weights, and minimising the negative entropy of the weights.

Bounded weights

The bounded weights regulariser simply constrains the values to a chosen set of bounds. It may be expressed as a regulariser in the objective function as

$$r(\mathbf{w}) = \begin{cases} 0 & \text{if } a \leq w_i \leq b, \quad \forall\, w_i \in \mathbf{w} \\ +\infty & \text{otherwise,} \end{cases} \tag{4.19}$$

or as an additional constraint.

Negative entropy

The negative entropy regulariser adds a penalty proportional to how far the sampling probabilities of the instances, determined by the weights, are from uniform (i.e. $\forall i,\ w_i = 1/N$). We do not want the sampling probabilities of the instances to be completely uniform since then the corpus would not be re-weighted at all, however, it may be desired that they are as uniform as possible while matching the target statistics. The balance between the loss function and the regulariser can be adjusted by scaling the regularisers penalty using a hyperparameter, denoted λ, as in the optimisation problem 4.12.

$$r(\mathbf{w}) = \sum_{i=1}^{N} w_i \log w_i \tag{4.20}$$

4.3.3 Convexity

Convexity plays an important role in the viability of EMM, allowing for solutions to optimisation problem 4.12 to be computed in an efficient and timely manner. Thus, there is a significant gain if it can be shown that the optimisation problem is convex. We give a brief explanation here as to why our formulation is a convex problem. For

completeness, the definitions of a convex function and a convex optimisation problem are given, as well as an outline of the arguments for the convexity of the loss functions and the suitability of the constraints.

Definition 4.3.1 (Convex set [33])**.** A set $S \in \mathbb{R}^n$ is a convex set if for any two points $x_1 \in S$ and $x_2 \in S$, we have that

$$\alpha x_1 + (1-\alpha)x_2 \in S, \quad \text{for all } \alpha \in [0,1]. \tag{4.21}$$

Definition 4.3.2 (Convex function [33])**.** The function $f : S \to \mathbb{R}$, where S is a convex set, is a convex function if for any two points $x_1 \in S$ and $x_2 \in S$, the following property holds:

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2). \tag{4.22}$$

Definition 4.3.3 (Convex optimisation problem [2])**.** An optimisation problem of the form

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{D}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \\ & h_i(\mathbf{x}) = 0, \quad i = 1, \ldots, p \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^n$, is convex if the objective function $f : D \subset \mathbb{R}^n \to \mathbb{R}$ is convex, the inequality constraints $g_i : \mathbb{R}^n \to \mathbb{R}$ are also convex, and the equality constraints $h_i : \mathbb{R}^n \to \mathbb{R}$ are affine.

We begin by confirming that each of the loss functions are convex functions. It is easily shown that the 2-norm, defined as $f(x) = \|x\|_2$, is convex. By the triangle inequality, we obtain the definition of a convex function, also given in Equation (4.22),

$$\|\alpha x_1 + (1-\alpha)x_2\|_2 \leq \alpha\|x_1\|_2 + (1-\alpha)\|x_2\|_2. \tag{4.23}$$

Since $g(x) = x^2$ is non-decreasing on the range f, then the squared 2-norm, defined by $g(f(x)) = \|x\|_2^2$, is also convex. As the least square loss function is essentially equivalent to the squared 2-norm, this implies that it is also convex.

As for the regulariser and loss functions with terms involving negative entropy, we may show that for $f(x) = x\log x$ is strictly convex on $x \in [0,1]$. Since $f'(x) = (1 + \log x)$ is a strictly increasing function, it implies that $f(x)$ is strictly convex. As the sum of convex functions is also convex [2], we conclude that the negative entropy is convex.

Furthermore, this allows us to use any combination of loss functions and regularisers in our problem and still maintain a convex objective function.

Equality loss may be defined as the equality constraint $\mathbf{F}^{\mathrm{T}}\mathbf{w} = \hat{\mathbf{f}}$ which is an affine function and therefore satisfies the conditions for convexity. A similar argument may also be used for inequality loss, as it can be defined as an inequality constraint. The weight constraint $\sum_i w_i = 1$ is affine, and $w_i \geq 0$ is linear, therefore these also satisfy the conditions for convexity.

4.4 Metrics

To measure the effectiveness of our re-weighting method, we propose two metrics: Jensen-Shannon (JS) distance and cross-classification. Using a known target dataset, summary statistics can be created and then used to generate synthetic data with EMM. The metrics can then be used to determine how similar a synthetic dataset is to the known target dataset.

JS distance measures the similarity between two distributions based on their relative entropy. Cross-classification gives a measure of how well the synthetic dataset can serve as a substitute for the target dataset in machine learning applications.

4.4.1 Jensen-Shannon distance

JS distance is based on relative entropy and it is the square root of Jensen-Shannon divergence. It is a symmetric and smoothed version of relative entropy that can be used as a distance metric. We choose JS distance over the relative entropy since it has some valuable properties that relative entropy does not; it is symmetric, always finite and obeys the triangle inequality. Hence, it falls within the mathematical definition of a metric [25].

Jensen-Shannon divergence is defined as

$$\mathrm{JS}_{\mathrm{div}}(P \,\|Q) = \frac{1}{2} D_{\mathrm{KL}}(P \,\|\, M) + \frac{1}{2} D_{\mathrm{KL}}(Q \,\|\, M), \tag{4.24}$$

where $M = \frac{1}{2}(P + Q)$. In our implementation, the logarithm within the relative entropy is taken to be base two. This ensures that JS divergence is bounded on $[0, 1]$ where 0 implies the distributions are equal almost everywhere [25].

4.4.2 Cross-classification

The second metric we consider is cross-classification, proposed in [11]. It uses the performance of machine learning classifiers to measure how well the re-weighted data mimics the statistical dependency structures of the the target data it is attempting to emulate.

There are two variations of the cross-classification metric. In the first, referred to as CrCl-RS, a classifier is trained on the target (real) dataset and the performance metrics for classification compared to hold-out test data from both the target and synthetic datasets. The second, called CrCl-SR, trains on the synthetic data and then tests on hold-out data from the target and synthetic datasets. This metric is useful in determining if models trained on data generated with EMM can be used to make reliable predictions on real-world data.

For both CrCl-RS and CrCl-SR, the general procedure for evaluating the metric is as follows:

1. Split the synthetic and target datasets into training and test sets.
2. Train on the target training dataset depending on choice of CrCl-RS or CrCl-SR.
3. Compute classification performance for both test datasets.
4. Compare classification performance by calculating ratio between them. One is considered to be a perfect score.

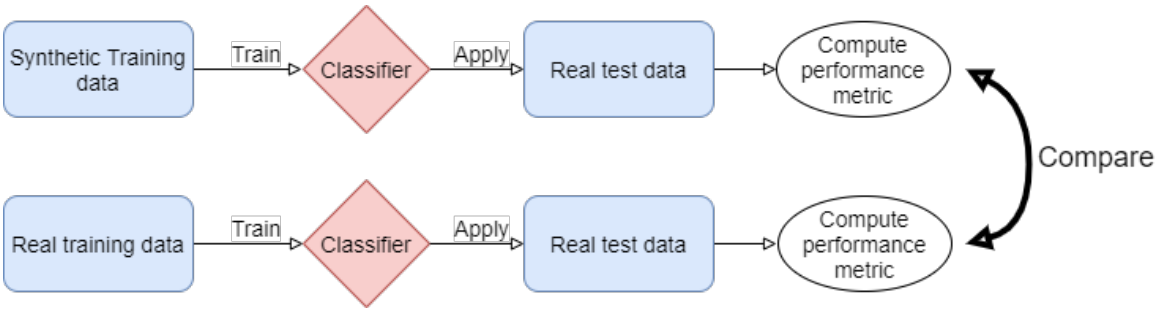

Figure 4.1: Flow diagram for CrCl-SR. First, a classifier is trained on synthetic data and another on real data. Then, both classifier predict on a test dataset of real data. Lastly, their performance is compared using a suitable metric.

Definition 4.4.1 (Classifier abbreviations)**.** We use the following abbreviations to describe what type of data a classifier is trained on and predicts/scored on.

- **RR** – Trained on real data, predicts on real data
- **RS** – Trained on real data, predicts on synthetic data
- **SS** – Trained on synthetic data, predicts on synthetic data
- **SR** – Trained on synthetic data, predicts on real data

Depending on the nature of the classification problem, the choice of the evaluation metric makes a considerable impact on how the model's performance is interpreted. In our work, we consider two scoring methods, accuracy and area under the curve (AUC) of the receiver operator characteristic (ROC) curve. For our purposes, they are defined for binary classification, where only two outcomes are being predicted.

Definition 4.4.2 (Accuracy score)**.** Accuracy score serves as a simple baseline metric, it is a measure of the number of correctly predicted samples over the total number of samples. In terms of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), it is defined as

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{4.25}$$

While accuracy score provides an intuitive indication of performance, it can often be a misleading measure, especially when dealing with imbalanced classes where the majority class represents the "normal" state and the minority class is the "abnormal" state. For example, in Section 6.3, where the aim is to predict whether or not a person has COVID-19, a much higher priority is placed on high TP and low FN classification. The dataset has a high class imbalance, with 94% of instances being labelled negative. Hence, if a model was trained with its performance based on accuracy, it is quite probable the classifier would be trained to predict all instances as negative. This model would achieve an accuracy of 94%, however, having zero TP predictions and all FN predictions.

For models trained on imbalanced datasets, we score their performance with a ROC curve and its corresponding AUC score. The ROC curve shows the trade-off between the true positive rate and the false positive rate.

Definition 4.4.3 (True positive rate (TPR))**.** The TPR of a classifier is the proportion of positive labels predicted correctly. It is calculated as the proportion of TP predictions in the number of positive cases. Mathematically, it is stated as

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.26}$$

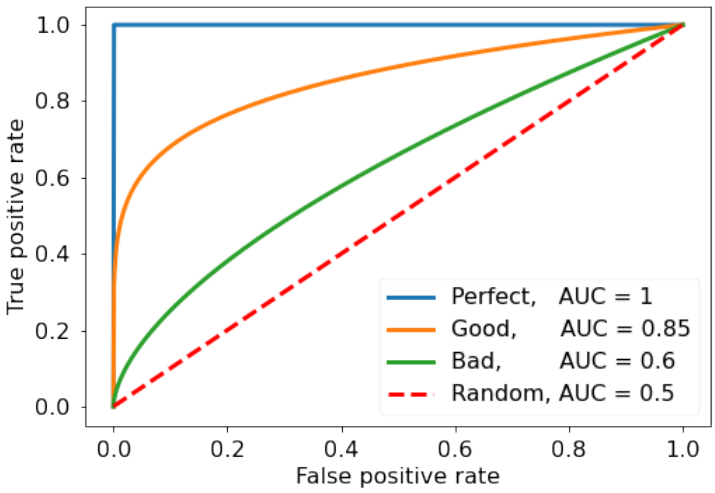

Figure 4.2: An example of different ROC curves with AUC scores for classifiers ranging from perfect to random (or no skill).

Definition 4.4.4 (False positive rate (FPR))**.** The FPR of a classifier is the proportion of negative labels predicted incorrectly. It is calculated as the proportion of FP predictions in the number of negative cases. Mathematically, it is stated as

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{4.27}$$

For classification problems such as disease screening, these provide important insights. A high TPR means a classifier performs well at correctly identifying cases who have the disease. However, this usually comes at the expense of an increased FPR, which implies that a classifier performs poorly at ruling out cases that do not have the disease. The ROC curve gives a visualisation of the trade-off between those two goals, as the decision threshold of the classifier is varied [10]. A low decision threshold may result in many TP cases, but also a large number of FP cases. Whereas, a high decision threshold would likely result in the opposite outcome.

Figure 4.2 gives an example of possible ROC curves with their corresponding AUC scores. A perfect classifier is one that can achieve a TPR of 1 and a FPR of 0, implying that it does not incorrectly classify any samples. For non-perfect classifiers, there is a trade-off; as the TPR increases, so too does the FPR. What separates good classifiers from bad classifiers is the rate at which this trade-off occurs, generally one desires a large increase in TPR for a relatively small increase in FPR.

In the legend of Figure 4.2, the AUC scores are given for each. AUC, which is the area under the ROC curve, provides an aggregate measure of the performance of a

classifier across all possible classification thresholds. It may be interpreted as the probability that a classifier will rank a randomly selected positive sample higher than a randomly selected negative one [10]. We use AUC as it provides a simple metric which may be used to easily compare classifiers, and offers significant improvements over accuracy score.

Chapter 5

Implementation

5.1 The EMM Python library

We have created an open-source library to implement EMM, as described in Chapter 4, using the Python programming language. The project is available for use and contribution at

`https://github.com/LucasKIJ/EMM`.

The aim of the library is to facilitate the seamless use of EMM with common Python data science libraries such as `pandas` [50] and `scikit-learn` [39]. The process of generating synthetic data with EMM is divided into four parts: pre-processing, finding the optimal weights, construction of the synthetic dataset, and evaluating metrics.

The main packages used are `numpy` [16] for linear algebra, `pandas` [50] for data manipulation, `cvxpy` [6] for convex optimisation, `scipy` [53] for special scientific functions, and `scikit-learn` [39] for construction machine learning classifiers.

Pre-processing

Pre-processing includes procedures that involve transforming raw data into a format that is ready to be used in the optimisation stage. Processes such as data importation, conversion, standardisation, one-hot encoding of categorical variables, and handling of missing data are needed to create a corpus dataset that can be used by the EMM optimisation function.

Optimisation in Python

There exists a large catalogue of optimisation packages in Python, including CPLEX [4], CVXPY [6], GUROBI [14] and MOSEK [30]. In our implementation, we choose to use the MOSEK optimiser through the CVXPY. MOSEK was chosen for its flexibility and efficiency in solving a wide variety of optimisation problems such as linear, quadratic, and mixed-integer programs [6]. CVXPY provides an easy and intuitive interface for formulating optimisation problems in Python.

For example, optimisation problem (4.12), defined with the least squares loss function and the entropy regulariser may be written as:

```
import cvxpy as cp
w = cp.variable(n); constrs = [cp.sum(w) == 1, w <= 1]
objective = cp.sum_squares(F @ w - f_target)
regulariser = - lam * cp.sum(cp.entr(w))
cp.Problem(cp.Minimize(objective + regulariser), constraints).solve()
```

Synthetic dataset generation

Synthetic datasets are created by calling the `generate_synth` function using a corpus defined as the pandas DataFrame `corpus`, and a dictionary of `marginal` objects. A `marginal` obejct is initialised by providing the target feature, the function of the target statistic, and the loss function (which takes an input of the target statistic value).

For example, to match the mean of the `"height"` feature to 185cm for class 0, and to 165cm for class 1, using the CorpusKL loss function and the entropy regulariser with the regulariser hyerparmeter $\lambda = 1$, we call the following code:

```
# Define marginal statistic objects and create dictionary
# with target class as keys and generate
marginals = {
    0: [marginal(feature="height", fun="mean",
                 loss=LeastSquaresLoss(185))],
    1: [marginal(feature="height", fun="mean",
                 loss=LeastSquaresLoss(185))],
}
data = generate_synth(corpus, marginals,
                regularizer=EntropyRegularizer(), lam=1)
```

Metric evaluation

The metrics JS distance and cross-classification have been implemented and designed to work with weighted data. By calling the function `classifier_metric`, performance can be compared between the target dataset and the synthetic data by separately training classifiers on each dataset using a grid search with cross-validation. The user has the freedom to choose what classifier, classifier parameters, scoring function, and cross-validation settings to use.

The function returns two classifiers: `target_clf` has been trained on the target data, and similarly `synthetic_clf` has been trained on the synthetic data. It also returns the scores from predictions on the target and test datasets, equivalent to the RR, RS, SS, and SR classifier scores given in Definition 4.4.1.

For example, suppose the known target dataset is the pandas DataFrame `target`, and the synthetic dataset is the DataFrame named `synthetic`. Then, to compute the cross-classification metric using logistic regression, scoring with ROC, and 5 fold cross-validation, we have the following code:

```
param_grid =  {"classifier" : [LogisticRegression()]}
(
 target_clf,
 synthetic_clf,
 scores
) = classifier_metric(target, synthetic, param_grid,
                      scoring=roc_auc_score, cv=5, return_models=True)
```

5.2 Weighted data in modelling

Once the weights have been calculated, there are two common methods for their use: weighted sampling and use in machine learning algorithms.

Weighted sampling

Since the weights describe the probability of selecting an instance $\mathbf{x}_i$ from its dataset. Hence, by sampling x_i we can create the distribution $X_1, \ldots, X_N$, we have

$$\mathbb{P}(X_i = \mathbf{x}_i) = w_i, \quad \text{for } i = 1, \ldots, N. \tag{5.1}$$

Once this distribution has been created, it may then be used in the same way as any other other dataset.

Incorporating weights into a loss function

Alternatively, many machine learning software packages, such as `scikit-learn`, allow for the direct incorporation of sample weights during training and scoring. In training, the weights are used to determine the severity of the penalty imposed by the loss function. For example, in many machine learning models, the weights may be integrated into the training minimisation problem with N training samples as

$$\min_{\theta} \quad \sum_{i=1}^{N} w_i \, \ell(f(\mathbf{x}^i;\ \theta), \hat{y}) \tag{5.2}$$

where $f(\,\cdot\,;\ \theta)$ is the model, $\hat{y}$ are the true label values, and ℓ is a chosen loss function.

Similarly, the weights can be included in scoring functions such as accuracy, f1-score, and AUC. The method of inclusion differs for each score. For those reliant on calculating true positives and so on, a weighted confusion matrix must be calculated. For accuracy, the calculation is relatively simple: it is the sum of the weights corresponding to the TP and TN samples divided by the sum of the weights.

Chapter 6

Results

To illustrate and test the theory discussed in Chapter 4, three experimental examples have been created using our method; EMM. In each example, summary statistics are created from a known target dataset, and then synthetic data is generated using EMM. The metrics defined in Section 4.4 are used to assess the quality of synthetic data versus the known target dataset. Of particular interest are the cross-classification scores which compare the performance of an RR classifier versus SR classifier (see Definition 4.4.1 for further details), as they help gauge the viability of training machine learning models on synthetic data.

6.1 Simulated cases

For a simple example of how this method may be applied to a continuous distribution, we generate a Gaussian mixture, which is the target distribution, i.e., the distribution that the re-weighted corpus is aiming to be statistically similar to. N_0 samples are generated from the Gaussian $\mathcal{N}(\mu_0, \sigma_0^2)$ are labelled 0 (or 'negative'), and N_1 are generated from the Gaussian $\mathcal{N}(\mu_1, \sigma_1^2)$ are labelled 1 (or 'positive'). We also create an unlabelled corpus by taking N_c samples from a skew normal distribution with mean μ_c, standard deviation σ_c, and shape a_c. In our example, we have the following parameters,

	a	μ	σ	N
Negative, 0	–	-0.3	0.35	5000
Positive, 1	–	0.2	0.3	5000
Corpus, c	2	-1	2	10000

Figure 6.1 shows the distribution of the corpus and target datasets. We re-weight one copy of the corpus data to have its mean equal to μ_0 and standard deviation equal to σ_0, and re-weight another copy to have mean μ_1 and standard deviation σ_1. Every instance is each re-weighted corpus is given the label of its corresponding target distribution. Hence, all instances in the corpus with mean μ_0 are given the label 0, and all instances in the second re-weighted corpus are given the label 1. We concatenate and then shuffle both re-weighted corpora, leaving us with a synthetic dataset with twice the number of instances as the corpus dataset.

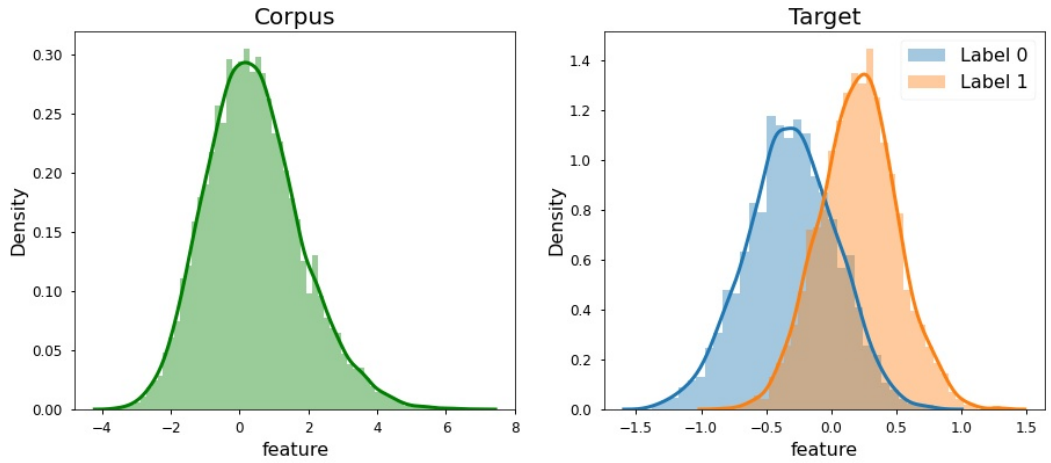

Figure 6.1: Probability density plots for the simulated data example. The unweighted corpus is given on the left and target datasets conditional on label value on the right.

A total of six synthetic datasets are created by re-weighting using two different loss functions and three different values for the regularisation hyperparameter. Least Squares loss and relative entropy loss (labelled Corpus KL) were used as loss functions and $\lambda = 0$, $\lambda = 0.01$, and $\lambda = 10$ as regularisation parameters. Using CVXPY and the MOSEK optimiser, the average time to find the optimum weights for both labels was 3.94 seconds for Corpus KL loss and 1.51 seconds for least squares loss.

In Table 6.1, we have given the relative errors between the synthetic data means and standard deviations, and those of the target dataset. We see that when $\lambda = 0$, that least squares loss achieves the lowest relative error across means and standard deviations. For Corpus KL loss, there is a negligible difference between $\lambda = 0$ and $\lambda = 1$. For $\lambda = 10$, both loss functions have relative errors of a similar order of magnitude.

Figure 6.2 shows the probability density plots for the different loss functions and values of λ. We see that as λ increases, the distribution conditional on labels of

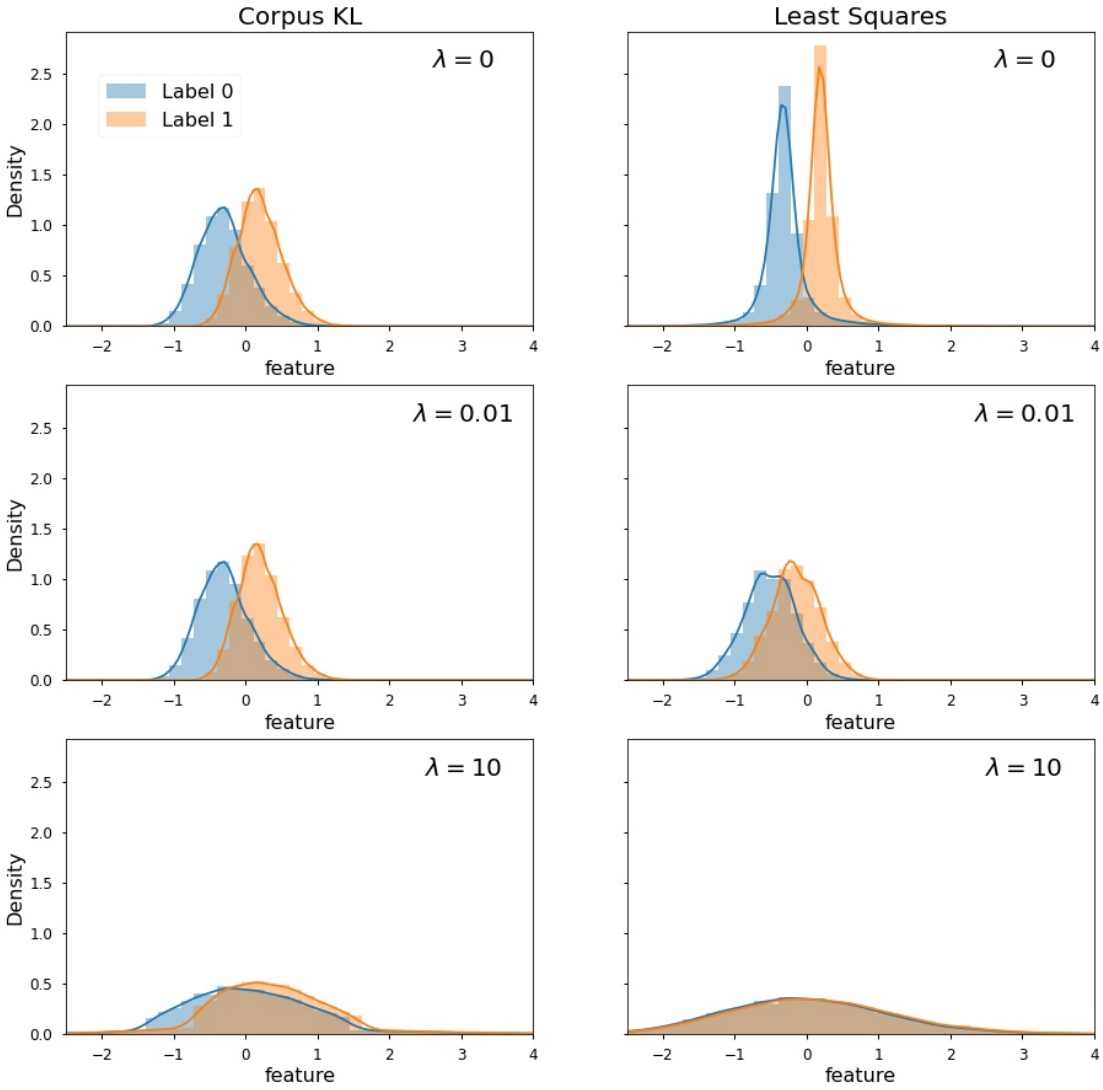

Figure 6.2: Re-weighted distributions for the simulated example under Corpus KL and Least squares loss function for a range of regulariser hyperparameters λ.

the re-weighted corpus converges towards the unweighted corpus. To quantify the difference between the target distributions and the synthetic data, we have calculated the JS distance for each synthetic dataset conditional on label. These values are given in Table 6.2, where the synthetic dataset created with the Corpus KL loss function and $\lambda = 0$ achieve the lowest score of 0.101 and 0.0989 for labels 0 and 1 respectively. This is a marked improvement from the JS distances between the target dataset and the unweighted corpus, given as 0.385 and 0.443.

Loss function	Regularisation strength	Relative error			
		μ_0	μ_1	σ_0	σ_1
Corpus KL	$\lambda = 0$	0.0067	0.0056	0.017	0.0299
	$\lambda = 0.01$	0.0076	0.0037	0.0322	0.0536
	$\lambda = 10$	1.31	0.823	1.71	2.00
Least square	$\lambda = 0$	3.2×10^{-8}	1.8×10^{-6}	1.1×10^{-4}	4.7×10^{-6}
	$\lambda = 0.01$	0.689	1.74	0.0275	0.149
	$\lambda = 10$	1.00	0.052	2.23	2.85

Table 6.1: Relative error between the target statistics and the synthetic data statistic for the simulated example.

Loss function	Regularisation strength	Jenson-Shannon distance	
		Negative	Positive
No re-weighting	n/a	0.385	0.443
Corpus KL	$\lambda = 0$	0.101	0.0989
	$\lambda = 0.01$	0.101	0.0992
	$\lambda = 10$	0.343	0.380
Least square	$\lambda = 0$	0.290	0.286
	$\lambda = 0.01$	0.233	0.435
	$\lambda = 10$	0.367	0.434

Table 6.2: Jenson-Shanon distance by class for simulated example

Loss function	Regularisation strength	Accuracy scores			
		RR	RS	SS	SR
Corpus KL	$\lambda = 0$	0.793	0.792	0.794	0.794
	$\lambda = 0.01$	0.782	0.784	0.794	0.794
	$\lambda = 10$	0.795	0.573	0.575	0.765
Least squares	$\lambda = 0$	0.790	0.897	0.899	0.790
	$\lambda = 0.01$	0.781	0.649	0.690	0.702
	$\lambda = 10$	0.786	0.509	0.503	0.643

Table 6.3: Corpus KL loss average accuracy values for Logistic regression, Decision Tree and Random Forest classifiers for the simulated example.

As a secondary metric, we have also trained three common models used for classification in machine learning: a logistic regression, a decision tree classifier, and a random forest classifier. We begin by splitting the target and synthetic data into training and test sets. Then, for each type of classifier, we train one model on the target training data and another model on the synthetic training data. The best hyperparameters are selected using a grid search with cross-validation. Lastly, we compute the accuracy scores for the following classifiers: RR, RS, SS, and SR. These abbreviations are defined in Definition 4.4.1. The average of the scores across the three classifiers are given in Table 6.3.

6.2 Self-testing on real data

To explore the validity of our method on a variety of real datasets, we have devised a self-testing technique, applicable to cases where a suitable corpus is absent. For a target dataset that we wish to compare EMM generated synthetic data against, we let the corpus be the target dataset but with the labels removed. Then, class-conditional summary statistics are generated from the target dataset. Using these summary statistics, we re-weight the corpus using EMM and evaluate the performance of our re-weighting using the metrics we described in Section 4.4.

For our example, we have chosen the PIMA Indian diabetes dataset collected by the National Institute of Diabetes and Digestive and Kidney Diseases [31]. The features of the dataset consist of the following variables: glucose concentration, diastolic blood pressure, skin thickness, insulin levels, BMI, diabetes pedigree function, and age. The aim is to use this variable to predict a binary outcome; whether the patient has diabetes (labelled 1 / negative) or not (labelled 0 / positive). It is worth noting that this dataset has a significant class imbalance with 65.1% of instances labelled 0 and 34.9% labelled 1.

We begin by calculating the summary statistics conditional on the outcome, these are given in Table 6.4. From these, we chose to re-weight with respect to the means corresponding to the features: glucose, BMI, and age. These features were chosen based on feature importance analysis given in [19]. The corpus is then re-weighted using the target statistics on their corresponding features with the least squares loss function and $\lambda = 0.01$. This was done in a similar way to that described in the previous simulated example, with the addition of the features being standardised to have a mean zero and standard deviation one. Since the corpus is relatively low

in dimension compared to the simulated example, the average training time is also significantly lower at 0.122 seconds taken to create the synthetic dataset.

Feature	**Corpus dataset**	**Diabetes dataset**	
		Negative ($n = 768$)	**Positive** ($n = 268$)
Pregnancies	3.84	3.30	4.87
Glucose	122	111	142
Blood pressure	72.4	70.9	75.3
Skin thickness	29.2	27.2	33.0
Insulin	156	130	207
BMI	32.5	30.9	35.4
Diabetes pedigree function	0.472	0.430	0.551
Age	33.2	31.2	37.1

Table 6.4: Means for each feature in the PIMA dataset, conditional on class.

Dataset	**Feature**	**Jenson-Shannon distance**	
		Negative	Positive
Unweighted corpus	Glucose	0.179	0.270
	BMI	0.111	0.183
	Age	0.119	0.204
Re-weighted corpus	Glucose	0.067	0.074
	BMI	0.045	0.104
	Age	0.069	0.122

Table 6.5: Jenson-Shanon distance between target and synthetic distribution conditional on class for the PIMA dataset.

Figure 6.3 shows a clear similarity between the target distribution and the synthetic data constructed from the re-weighted corpus. However, the distributions for the underrepresented class, label 1, appear not to match as well as those from label 0. This observation is further quantified by the JS distances given for the distributions in Table 6.4, where we see that JS distances between the distributions for label 1 are all higher than those for label 0.

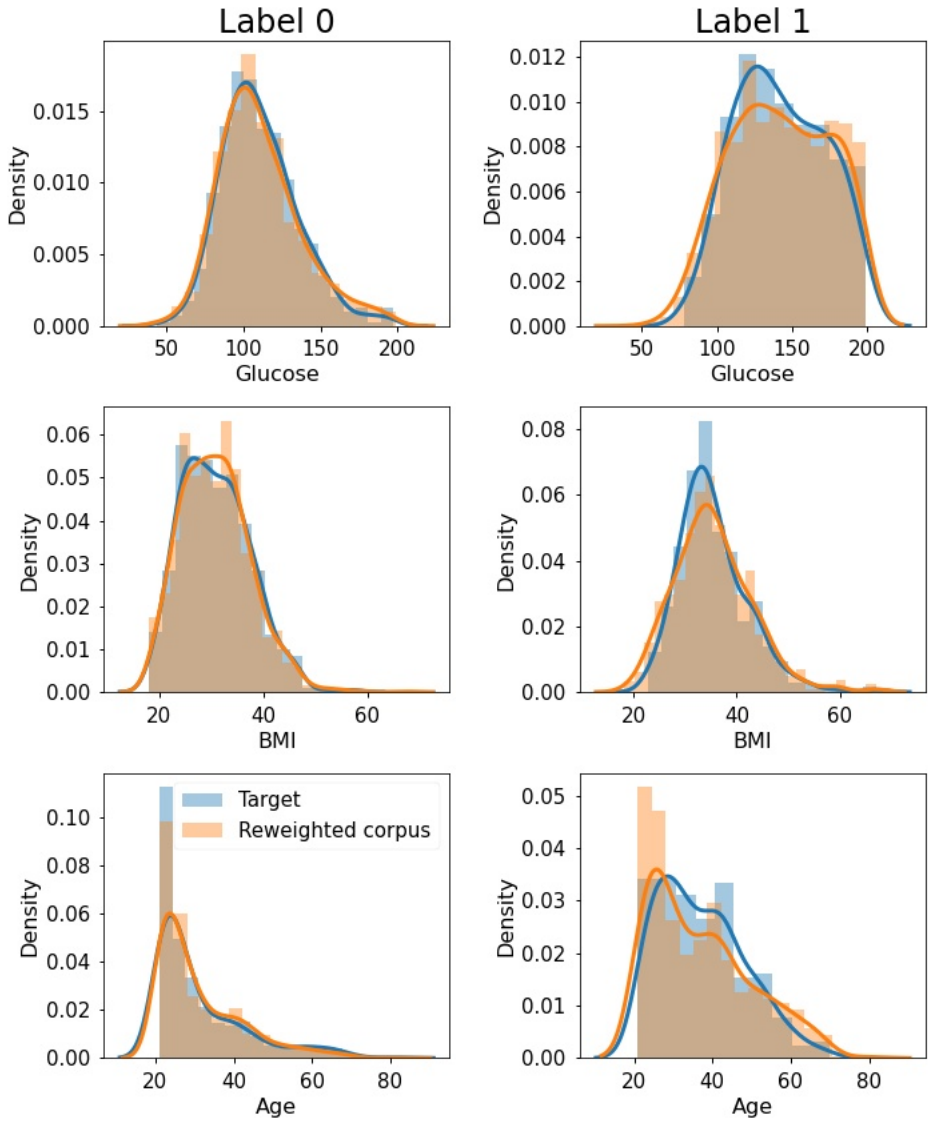

Figure 6.3: Comparison of the target (blue) versus re-weighted corpus (orange) distribution for the glucose, BMI, and age feature in the PIMA dataset.

		ROC AUC Score			
Parameters	**Classifier**	RR	RS	SS	SR
Least square loss,	Logistic Regression	0.844	0.793	0.796	0.830
$\lambda = 0.01$	Decision Tree	0.718	0.742	0.621	0.802
	Random Forest	0.775	0.746	0.663	0.756

Table 6.6: Classifier AUC ROC scores for PIMA predictions with regulariser value of $\lambda = 0.01$.

The cross-classification scores are given in Table 6.6 where AUC was used as the classifier metric as it provides a better indication of performance on imbalanced datasets. Similarly to the simulated data example, we trained logistic regression, decision tree, and Random Forest classifiers on both the target and synthetic datasets to provide RR, RS, SS, and SR scores. All three classifiers perform similarly on real and synthetic data, with similar scores being achieved in RR and SR metrics.

6.3 An application to COVID-19 prediction

As an example for potential practical uses for EMM, we have re-weighted a medical corpus to match summary statistics from another medical study. We use the Hospital admissions dataset as the corpus, which was collected from the enterprise data warehouse, using SQL queries to extract relevant medical data by Hong et al. [17]. The dataset contains a total of 972 variables and 557,901 instances, providing a rich corpus of socio-demographic and medical variables to work with that frequently overlap with those recorded in other medical studies.

The target dataset was obtained from the recently published paper by Zoabi et al. [59], titled "Machine learning-based prediction of COVID-19 diagnosis based on symptoms". Its original source was the Israeli Ministry of Health, who publicly released data of individuals who were tested for COVID-19 via RT-PCR assay of a nasopharyngeal swab [49].

The target dataset contains various clinical attributes about the individual being tested, such as age 60 years or above, sex, and five common COVID-19 symptoms. Each instance has a class label, the results of their COVID-19 test. There exists a significant class imbalance , with only 5.36% of the 274,956 instances being positive cases.

In Table 6.7, the summary statistics for the corpus and target datasets are provided. The statistics for the target are given in a similar fashion to how they were presented in the original publication. All features were binary encoded, noting that gender has the label 1 for males and label 0 for females. The mean value of a binary variable is equal to the proportion of instances with a 1-encoding.

Using the summary statistics in Table 6.7, we re-weight using the 'positive' proportion as the target means. The least squares loss function and a regulariser value of $\lambda = 0.01$ were used. The regulariser value was chosen as it performed best with respect to the

Feature	Flag	Corpus dataset	COVID-19 dataset	
			Negative ($n = 260,227$)	**Positive** ($n = 14,729$)
Age 60+	True	31.4	16.9	19.2
	False	68.6	83.1	80.8
Gender	Male	44.8	49.5	55.4
	Female	55.2	50.5	44.6
Cough	True	2.3	13.5	44.8
	False	97.7	86.5	55.2
Fever	True	1.4	6.1	37.8
	False	98.6	93.9	62.2
Sore throat	True	1.3	0.1	10.4
	False	98.7	99.9	89.6
Shortness of breath	True	4.4	0.1	7.9
	False	95.6	99.9	92.1
Headache	True	0.2	0.06	15.2
	False	99.8	99.94	84.8

Table 6.7: Summary statistics for overlapping features in COVID-19 target dataset and corpus dataset. They indicate the percentage proportion of 'positive' occurrences versus 'negative' occurrences.

classifier metric. As before, we re-weight the corpus once with the target means corresponding to label 0 and once more with the target means for label 1. Then, the two re-weighted corpora are concatenated to form the final synthetic dataset, which has twice as many instances as the corpus. It took 165.3 seconds to compute the weights for the synthetic dataset using CVXPY with the MOSEK optimiser.

The proportion of features with label 1 for the synthetic dataset and the error between those values and the target dataset's are given Table 6.8. The errors for the COVID-19 negative population are all less than 1%, implying the corpus has been re-weighted to accurately represent the distribution of target values seen in the summary statistic table. For COVID-19 positive values, the errors are slightly greater, with the largest being 1.51%.

Following the machine learning approach described in [59], a gradient boosted decision trees classifier, specifically the LightGBM package [21], was used to compute our cross-classification metric. Due to the large class imbalance in the target dataset, we have used ROC curves to compare the performance of a classifier trained on real data and

tested on real data, and one trained on synthetic and tested on real. Figure 6.4 shows the ROC curves and in the legend the AUC scores are also given.

	Synthetic apportionment (error)			
Feature	(%) Negative		(%) Positive	
Age 60+	17.2	(−0.39)	19.5	(−0.31)
Gender	49.4	(+0.09)	55.3	(+0.20)
Cough	12.5	(+0.92)	43.3	(+1.47)
Fever	5.42	(+0.66)	36.3	(+1.51)
Sore throat	0.685	(−0.55)	10.0	(+0.32)
Shortness of breath	0.897	(−0.75)	7.48	(+0.42)
Headache	0.179	(−0.12)	13.7	(+1.46)

Table 6.8: Proportion of 'positive' instances and errors for each feature in the re-weighted corpus.

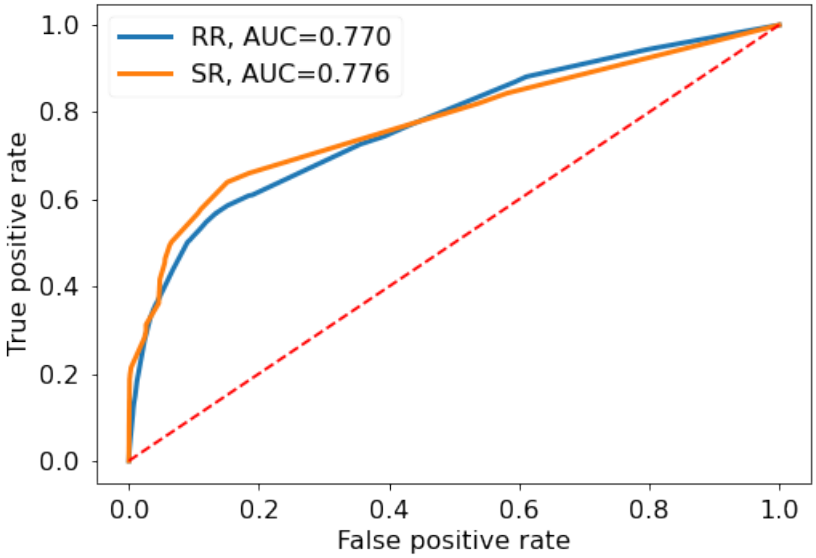

Figure 6.4: ROC curves, showing performance of the model trained on real data and predicting on real data (RR in blue), and the model trained on synthetic data and predicting on real (SR in orange).

Chapter 7

Discussion

The difficulties surrounding personal data access for research has necessitated the use of synthetic data generation techniques. However, there exist few methods that are capable of generating realistic data without access to the original data source that they seek to replicate. In Chapter 4, we have formulated a framework that aims to generate realistic synthetic data from summary statistics, without access to the data these statistics were calculated from. Using the implementation of EMM, described in Chapter 5, several experimental examples have been constructed to test the quality of our method.

The examples of the results section sequentially build in complexity to demonstrate the robustness of EMM when working with a wide variety of data types and sizes. The first example provides an illustration of the way different parameters such as loss function or regulariser impact the effectiveness of EMM. It also confirms that the the theory discussed in Chapter 4 can be applied successfully to simple continuous distributions, providing a suitable foundation to build upon for use with real patient records and statistics.

The PIMA diabetes example, described in Section 6.2, gives two key insights. Firstly, it showcases the "self-testing" procedure, a methodology for testing data types before an adequate corpus has been found. Secondly, the promising scores from the evaluation metrics, give further evidence that EMM can be successfully applied to multivariate non-normal data.

The final example demonstrates how EMM may be used to solve real-world problems. The hospital admission dataset [17], which pre-exists the COVID-19 pandemic, was re-weighted and used to successfully train classifiers to have the same performance as

those trained on real COVID-19 patient data. This shows a significant potential for the use of EMM to solve problems where the available data is limited by repurposing of data previously seen to be inapplicable.

Whilst not discussed explicitly in the results section, all examples can be said to produce realistic synthetic data where individual samples indistinguishable from its real counterpart. This is possible due to the availability of a corpus with the same features as the target dataset.

7.1 Simulated results

Section 6.1 presents a simple example where the target dataset consists of data generated from two Gaussian with different means and standard deviation. Instances labelled 0 were generated from the Gaussian distribution $\mathcal{N}(-0.3, 0.35^2)$, and those labelled 1 from the distribution $\mathcal{N}(0.2, 0.3^2)$. The simulated corpus was generated from a skewed Gaussian, with skew, mean, and standard deviation different from those of the target dataset. One of the key benefits of this example is that it provides a visual explanation of how EMM transforms the corpus distribution through re-weighting and allows for easy interpretation of both metrics.

The cross-classification scores of greatest interest are the RR and SR scores, as the ratio between them indicates the potential for using synthetic data to train models for prediction on real data. Due to the simple nature of this simulated example, classifiers are able to perform well even when trained on synthetic data that is almost inseparable, such as the data generated with large regulariser hyperparameter values. It also appears that the Corpus KL loss function is less sensitive to changes in the regulariser hyperparameter than least squares, maintaining a RR to SR ratio close to 1 for all hyperparameter values.

7.2 PIMA diabetes

The PIMA diabetes dataset [31] is a commonly used as an example dataset in machine learning. Offering a range of non-normally distributed variables, it provides a more challenging re-weighting task than the simulated data example. As the corpus used is the unlabelled target dataset, the example also illustrates the power of EMM as a selective procedure.

The JS distance scores, given in Table 6.5, show significant improvement in "likeness" of the class-conditional distributions of the target features over the unweighted corpus. This provides strong evidence that the re-weighting procedure is effectively allocating larger weights to those samples that are more likely to be members of the target class. We also note that JS distances for the minority class (label 1) are higher than those of the majority class (label 0). Since the corpus is the unlabelled target dataset, we believe this is a result of the regulariser having a greater effect when re-weighting for minority classes since larger weights must be allocated to fewer samples to match the target statistics, and consequently the weights have a greater entropy.

The cross-classification scores indicated that the synthetic data is a suitable substitute for the training of classifiers. The RR and SR scores given in Table 6.6, show that prediction performance on real data is equal for classifiers trained on synthetic data versus those trained on real data. This also provides strong evidence that our synthetic data mimics the core statistical attributes of the real data and asserts that we have achieved our aim of creating realistic data that performs well under machine learning use cases.

7.3 COVID-19 data

While the two previous examples have outlined the ability of EMM to produce synthetic data from summary statistics that are representative of the target distributions, they have not been examples of real-world applications. Using the hospital admissions corpus [17], collected in 2017, and the COVID-19 summary statistics [59], collected in 2020, we were able to construct a practical example for the use of EMM.

Using the summary statistic table computed from the original covid data of [59], were where able to create two classifiers, one trained on the original data and one trained on the synthetic data trained on the summary statistics of the original data. The ROC curves of Figure 6.4 show not only that both classifiers are able to achieve a TPR of approximately 70% keeping the FPR at around 20%, but that in general their predictive powers on real data are virtually indistinguishable. Note, while these results are encouraging, further stress testing is needed to understand EMM's limitations, in particular with respect to other synthetic generation methods.

As mentioned in Chapter 2, in the early stages of the COVID-19 pandemic, a minimal amount of data is freely available. However, many studies exist that published statistical tables conditional on where COVID-19 was present in a patient. These results

indicated that EMM may be a viable method to create alternative data for use in the training of machine learning classifiers in situations such as the one described above. This could allow for improved disease detection and screening in early stage disease discovery and control.

As a secondary insight, it is worth noting that the corpus used was collected two years before the COVID-19 pandemic began. The performance of classifiers trained on synthetic data generated by EMM suggests that it may be able to provide a valuable method to repurpose data that previously may be seen as inapplicable to classification problems.

7.4 Limitations

EMM is constrained by access and availability of a suitable quality corpus. Without a corpus with features that overlap with the target dataset, it is impossible to construct synthetic data of the same form as the target dataset. In cases where the corpus only contains a subset of features from the target dataset, then it cannot generate examples containing all features, thus reducing its utility. For the corpus to be suitable, it must also be large enough so that that the target distribution characteristics can be recovered through re-weighting.

The method is also limited by the similarity in distribution between the corpus and the target dataset. When the only known information available about the target distribution are summary statistics, the shape or type of target distribution often cannot be inferred. Hence, if the corpus follows a significantly different distribution to the target, then it is likely that the reweighted distribution may not resemble the target distribution accurately.

7.5 Future research and extensions

While the results of this paper demonstrate that EMM provides a promising method for synthetic data generation, to truly understand its effectiveness, it must be compared with other methods. In further research, we believe comparisons with methods such as multi-variate Gaussian sampling would aid in determining the distinct advantages of EMM. In particular, it would be useful to know if the latent relationships that exist between variables in the corpus lead to the creation of higher quality synthetic data.

In this paper, all metrics were designed to be used in tests where the target data is available, and therefore the synthetic data can be compared to it. To increase EMM's viability for use in real-world situations, future research should consider exploring metrics for measuring the quality of the synthetic data when the target dataset is unavailable. We believe further investigation of the distribution of the weights would likely yield promising results in relation to this.

Further testing of matching central moments using the approximation described in Section 4.2 is also suggested. As this approximation relies on the target mean and the mean of the synthetic data being approximately equal, it would be interesting to explore how parameters, such as the regulariser hyperparameter, effect its accuracy. If this method is shown to be suitably robust, then to the best of our knowledge, it would be the only weighting method that can match higher order moments.

Chapter 8

Conclusion

The aim of this research was to formulate a framework for the generation of synthetic data using class-conditional summary statistics created from an unavailable, private dataset. Our method was designed with two objectives: first, it must generate realistic samples which are indistinguishable from the private dataset, and second, it must do so without access to the private dataset. In our review of related work, given in Chapter 3, most methods, especially those published in the field of synthetic data generation, could not achieve both goals simultaneously. By expanding the search to include survey weighting methods, we have found that with some modifications, these methods can be used as synthetic data generators that achieve the two objectives described above.

In Chapter 4, we have adapted the recent (2021) and unknown (zero citations) work on survey weighting of Barrat et al. [1]. By extending it to match higher order moments, as well as designing a framework for re-weighting an unlabelled corpus to match multiple class-conditional statistics, we have created a method for synthetic data generation, called Entropic Marginal Matching.

The main contribution of this work is the use of weighting methods in conjunction with a corpus dataset to generate synthetic data, which is formulised as EMM. Since the corpus is composed of real samples with the same features of the private dataset, the synthetic samples generated by EMM are guaranteed to be realistic.

The effectiveness of our implementation of EMM in Python was tested on simulated, PIMA Indians diabetes, and COVID-19 datasets, which can be found in Chapter 6. Under the metrics defined in Section 4.4, the results of each example provide significant evidence that EMM is a viable method for synthetic data generation. Across

all three examples, classifiers trained on synthetic data performed equally to those trained on real data. This provides further evidence that synthetic data generated with EMM can serve as a suitable substitute for training classifiers that can then be used as a predictor on real data.

While this research was originally conceived to address the lack of data availability in healthcare, we believe EMM has a much wider potential for application. It provides a method for repurposing data to study novel problems, as demonstrated by the COVID-19 example, and as a way to infer far more than previously possible from the limited information provided in summary statistics. We hope it allows future researchers to make greater advances by enabling further use of published results, even when the raw data is unavailable.

References

[1] S. Barratt, G. Angeris, and S. Boyd, *Optimal representative sample weighting*, Statistics and Computing, 31 (2021), pp. 1–14.

[2] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, (2004).

[3] M. Bozkurt and M. Harman, *Automatically generating realistic test input from web services*, Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE), (2011), pp. 13–24.

[4] CPLEX, *V12. 1: User's manual for cplex*, International Business Machines Corporation, 46 (2009), p. 157.

[5] W. E. Deming and F. F. Stephan, *On a Least Squares Adjustment of a Sampled Frequency Table When the Expected Marginal Totals are Known*, The Annals of Mathematical Statistics, 11 (1940), pp. 427 – 444.

[6] S. Diamond and S. Boyd, *CVXPY: A Python-embedded modeling language for convex optimization*, Journal of Machine Learning Research, 17 (2016), pp. 1–5.

[7] Domo, Inc., *Data never sleeps 8.0*, (2020). Available at `https://www.domo.com/learn/infographic/data-never-sleeps-8`.

[8] K. El Emam, E. Jonker, L. Arbuckle, and B. Malin, *A Systematic Review of Re-Identification Attacks on Health Data*, PLoS one, 6 (2011), pp. 1–12.

[9] European Commission, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*, OJ, L 119 (2016), pp. 1–88.

[10] T. Fawcett, *An introduction to ROC analysis*, Pattern recognition letters, 27 (2006), pp. 861–874.

[11] A. Goncalves, P. Ray, B. Soper, J. Stevens, L. Coyle, and A. P. Sales, *Generation and evaluation of synthetic patient data*, BMC medical research methodology, 20 (2020), pp. 1–40.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, Advances in neural information processing systems, 27 (2014).

[13] W.-j. Guan, Z.-y. Ni, Y. Hu, W.-h. Liang, C.-q. Ou, J.-x. He, L. Liu, H. Shan, C.-l. Lei, D. S. Hui, et al., *Clinical characteristics of coronavirus disease 2019 in china*, New England journal of medicine, 382 (2020), pp. 1708–1720.

[14] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, (2020).

[15] M. G. Hansson, H. Lochmüller, O. Riess, F. Schaefer, M. Orth, Y. Rubinstein, C. Molster, H. Dawkins, D. Taruscio, M. Posada, et al., *The risk of re-identification versus the need to identify individuals in rare disease research*, European Journal of Human Genetics, 24 (2016), pp. 1553–1558.

[16] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, *Array programming with NumPy*, Nature, 585 (2020), pp. 357–362.

[17] W. S. Hong, A. D. Haimovich, and R. A. Taylor, *Predicting hospital admission at emergency department triage using machine learning*, PloS one, 13 (2018), p. e0201016.

[18] Insee, *Histoire de vie*, (2003). Available at `https://www.insee.fr/fr/statistiques/2532244`.

[19] J. Y. Ahn, *PIMA Indian diabetes : feature importance analysis*, Kaggle, (2018). Available at `https://www.kaggle.com/jy93630/pima-indian-diabetes-feature-importance-analysis`.

[20] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-Wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, *MIMIC-III, a freely accessible critical care database*, Scientific data, 3 (2016), pp. 1–9.

[21] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, *LightGBM: A Highly Efficient Gradient Boosting Decision Tree*, Advances in Neural Information Processing Systems, 30 (2017).

[22] D. P. Kingma and M. Welling, *Auto-Encoding Variational Bayes*, 2014.

[23] R. Kohavi and B. Becker, *The Adult Dataset*, (1996). Available at `https://archive.ics.uci.edu/ml/datasets/adult`.

[24] S. Kullback and R. A. Leibler, *On information and sufficiency*, The annals of mathematical statistics, 22 (1951), pp. 79–86.

[25] J. Lin, *Divergence measures based on the shannon entropy*, IEEE Transactions on Information theory, 37 (1991), pp. 145–151.

[26] MacArthur Foundation Research Network on Successful Midlife Development, *Midlife Development in the U.S. (MIDUS)*, (1996). Available at `https://www.midus.wisc.edu/index.php`.

[27] D. J. MacKay and D. J. Mac Kay, *Information theory, inference and learning algorithms*, Cambridge university press, (2003).

[28] S. McLachlan, K. Dube, T. Gallagher, J. A. Simmonds, and N. Fenton, *Realistic Synthetic Data Generation: The ATEN Framework*, (2019), pp. 497–523.

[29] A. Mercer, A. Lau, and C. Kennedy, *For weighting online opt-in samples, what matters most?*, Pew Research Center, (2018).

[30] MOSEK ApS, *MOSEK Optimizer API for Python*, (2021).

[31] National Institute of Diabetes and Digestive and Kidney Diseases, *PIMA Indians Diabetes dataset*, (1990). Available at `https://archive.ics.uci.edu/ml/datasets/diabetes`.

[32] S. I. Nikolenko, *The Early Days of Synthetic Data*, Springer International Publishing, Cham, 2021, pp. 139–159.

[33] J. Nocedal and S. Wright, *Numerical optimization*, Springer Science & Business Media, (2006).

[34] P. Ohm, *Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization*, UCLA L. Rev., 57 (2009), p. 1701.

[35] S. P. Parker, *McGraw-Hill dictionary of scientific and technical terms*, McGraw-Hill, Inc., (1994).

[36] Parliament of Canada, *Personal Information Protection and Electronic Documents Act (PIPEDA), S.C. 2000, c. 5*, (2000).

[37] Parliament of the United Kingdom, *Data Protection Act 2018 c.12*, (2018).

[38] J. Pearl, *Bayesian networks: A model cf self-activated memory for evidential reasoning*, Proceedings of the 7th conference of the Cognitive Science Society, University of California, Irvine, CA, USA, (1985), pp. 15–17.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[40] K. Rahimi, *Pharmacological blood pressure lowering for primary and secondary prevention of cardiovascular disease across different levels of blood pressure: an individual participant-level data meta-analysis*, (2021).

[41] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, *Generative Adversarial Text to Image Synthesis*, (2016).

[42] Research support, University of Oxford, *Information about ethics and approvals for research involving human participants*, (2021). Available at `https://researchsupport.admin.ox.ac.uk/governance/ethics/faqs-glossary/faqs`.

[43] L. Rocher, J. M. Hendrickx, and Y.-A. De Montjoye, *Estimating the success of re-identifications in incomplete datasets using generative models*, Nature communications, 10 (2019), pp. 1–9.

[44] A. Salim, *Synthetic Patient Generation: A Deep Learning Approach Using Variational Autoencoders*, (2018).

[45] H. Surendra and H. Mohan, *A Review of Synthetic Data Generation Methods for Privacy Preserving Data Publishing*, International Journal of Scientific & Technology Research, 6 (2017), pp. 95–101.

[46] S. Swaminathan, K. Qirko, T. Smith, E. Corcoran, N. G. Wysham, G. Bazaz, G. Kappel, and A. N. Gerber, *A machine learning approach to triaging patients with chronic obstructive pulmonary disease*, PloS one, 12 (2017), p. e0188532.

[47] S. Swaminathan, B. Toro, N. Wysham, N. Mark, S. Ramanathan, J. Morrill, and C. Landon, *CovidX: Remote Screening, Surveillance, Triage, and Management of Novel Coronavirus*, (2021). Available at `https://covidx.vironix.ai`.

[48] L. Sweeney, A. Abu, and J. Winn, *Identifying Participants in the Personal Genome Project by Name*, Data Privacy Lab, IQSS, Harvard University, (2013).

[49] The Israelie Ministry of Health, *COVID-19 Government data*, (2018). Available at `https://data.gov.il/dataset/covid-19`.

[50] The pandas development team, *pandas-dev/pandas: Pandas*, (2020).

[51] United States Census Bureau, *Public Use Microdata Samples (PUMS)*, (2020). Available at `https://www.census.gov/programs-surveys/acs/microdata.html`.

[52] G. Ursin, S. Sen, J.-M. Mottu, and M. Nygård, *Protecting privacy in large datasets—first we assess the risk; then we fuzzy the data*, Cancer Epidemiology and Prevention Biomarkers, 26 (2017), pp. 1219–1224.

[53] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, et al., *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods, 17 (2020), pp. 261–272.

[54] J. Walonoski, M. Kramer, J. Nichols, A. Quina, C. Moesel, D. Hall, C. Duffett, K. Dube, T. Gallagher, and S. McLachlan, *Synthea:*

An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record, Journal of the American Medical Informatics Association, 25 (2017), pp. 230–238.

[55] M. A. Whiting, J. Haack, and C. Varley, *Creating Realistic, Scenario-Based Synthetic Data for Test and Evaluation of Information Analytics Software*, Proceedings of the 2008 Workshop on BEyond Time and Errors: Novel EvaLuation Methods for Information Visualization, (2008).

[56] G. Wiederhold and B. Lee, *A rule-based software test data generator*, IEEE Transactions on Knowledge & Data Engineering, 3 (1994), pp. 108–119.

[57] A. Yale, S. Dash, R. Dutta, I. Guyon, A. Pavao, and K. P. Bennett, *Generation and Evaluation of Privacy Preserving Synthetic Health Data*, Neurocomputing, 416 (2020), pp. 244–255.

[58] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin, *Adversarial feature matching for text generation*, vol. 70 of Proceedings of Machine Learning Research, (2017), pp. 4006–4015.

[59] Y. Zoabi, S. Deri-Rozov, and N. Shomron, *Machine learning-based prediction of covid-19 diagnosis based on symptoms*, npj digital medicine, 4 (2021), pp. 1–5.