Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter now, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Yunfan Jiang April 12, 2022

Synthetic Generation of Datasets With Complex Attributes

by

Yunfan Jiang

Dorian Arnold
Adviser

Department of Computer Science

Dorian Arnold, Ph.D.
Advisor

Michelangelo Grigni, Ph.D.
Committee Member

Ting Li, Ph.D.
Committee Member

2022

Synthetic Generation of Datasets With Complex Attributes

By

Yunfan Jiang

Dorian Arnold
Adviser

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements for the degree of
Bachelor of Science with Honors

Department of Computer Science

2022

Abstract

Synthetic Generation of Datasets With Complex Attributes

By Yunfan Jiang

Synthetic data generation has been applied usefully in many domains, when real data is unavailable, for example, due to logistical, technical, or policy issues, like privacy or security. However, there is a lack of study on synthetic data generations of complex datasets that contain temporal features or categorical features with a large number of labels. This thesis focuses on identifying a robust generative method to produce accurate results for such datasets. We studied a university student dataset with the DataSynthesizer tool. We developed and tuned appropriate data preprocessing, data encodings and model configurations to improve the quality of the synthesized results. Using both standard visualizations and student-specific statistics, we show how our approach can achieve feasible synthetic results that are as good as those from less complex datasets.

Synthetic Generation of Datasets With Complex Attributes

By

Yunfan Jiang

Dorian Arnold
Adviser

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements for the degree of
Bachelor of Science with Honors

Department of Computer Science

2022

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data analytics, including statistical models, have become important in many domains, such as cancer prediction [1]. High quality data are critical for the analytics to be useful. For instance, data analysis tasks like classification and prediction may be inaccurate without enough quality data to analyze.

In many cases, enough high quality data may not be available due to privacy or security issues. For example, in the medical area, drug response prediction is challenging due to data unavailability [1]. To deal with data unavailability, synthetic data generation has become a viable technique. Synthetic data generation is the process of creating artificial data that have the same statistical properties as a real dataset, making it indistinguishable from real data. It has several benefits, such as increasing data availability and protecting privacy. Hence, it has been applied in many areas, such as renewable energy resources [11], traffic sign recognition [13], and medical research [6]. These works demonstrate good performance on datasets with either continuous attributes like wind speed [11] or categorical attributes that have a small set of values [13]. However, due to limited studies, their performance on more complex datasets, such as datasets with temporal features or categorical features with a larget set of values, is unknown. Though [6] claims to study their model on

a large-set dataset, the maximum count of categories is still less than 200, whereas more complex data, for example our university student dataset, can have categorical attribute sets greater than 3000.

This work studied the synthetic generation of a dataset of university student course records. This dataset satisfies our research interest, since it contains a course name attribute, which is a categorical attribute with a large and specific set of values. It also includes a term attribute, which is a temporal feature. As a result, this project aimed to identify a robust generative method that effectively synthesizes complex datasets with categorical attributes that have large set size, and we tested our method with a student course record dataset.

Our general approach is based on an existing generation tool, DataSynthesizer [12], that uses a Bayesian network. We tuned the tool's configuration and incorporated other enhancements to address issues raised by our dataset. We demonstrate that our finalized model is effective according to both standard visualization metrics and student-specific statistics. Summarily, we developed a set of techniques and enhancements that allowed the tool to generate accurate results for complex datasets.

In this thesis, Chapter 2 introduces relevant background and related literature. Chapter 3 describes the methodology behind the data generation process, including generic computer science models, evaluation strategies, and an established tool to help synthesize data. Chapter 4 details our experiments, dataset, testing methodology, experimental setup, results, and result analysis. Finally, in Chapter 5, we conclude with a summary of our work and opportunities for future improvements.

Chapter 2

Background & Related Work

A general synthetic data generation process involves three main components: a real dataset as an input, a generation model, and evaluation methods. The input dataset will be applied into a generation model, and the model will produce a synthetic dataset that will be evaluated according to some preset metrics. This process is also demonstrated in Figure 2.1. The process may be iterative if evaluation results are unsatisfactory, with parameter tuning and model adjustments incorporated into the generation workflow.

2.1 Input & Synthetic Datasets

In this section, we introduce some definitions related to both input and synthetic datasets that are involved in the generation process.

Our dataset comprises courses records, which are represented as rows, and each column represents an attribute. We used the terms "attribute" and "feature" interchangeably to name the variable that each column of the dataset represents, with the assumption that all values of an attribute are the same category or "type" of data. For example, if "age" is an integer attribute, every unique instance will be an integer rather than some other type such as a string.

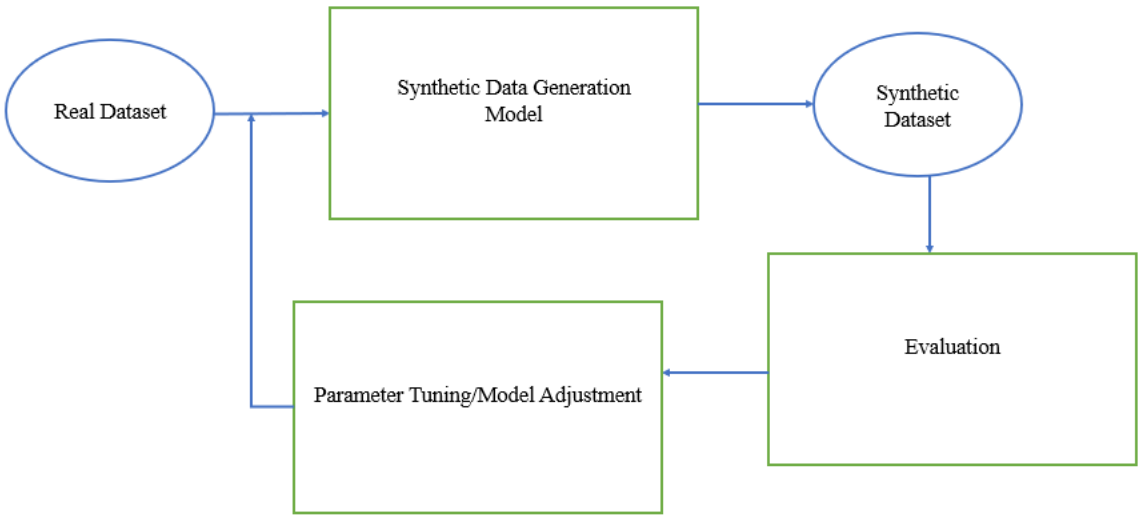


Figure 2.1: The general synthetic data generation process: a real dataset is input into a generation model to produce synthetic data, which is then evaluated for quality. This process can include iterative parameter tuning and model adjustments until the synthetic data quality is evaluated to be good.

Second, we used the term "domain size" [12] to represent the size of the set of legal values of a categorical attribute. For instance, if an attribute called "gender" only has two distinct values, its domain size will simply be 2. Since each attribute has a different domain size, the "largest domain size" of a dataset is further defined as the one that has the largest value among all attributes' domain sizes.

Third, in synthetic data generation, categorical and non-categorical features are treated differently. If an attribute is categorical, the distribution percentage of each category in synthetic datasets should be close to that in real datasets, making the generation process more complicated. By contrast, rather than evaluating distributions of each value, non-categorical features will be evaluated based on each group that includes a range of values, so they are evaluated less strictly. This is reasonable because non-categorical attributes do not have a fixed value set, so it is not significant to learn each value's distribution. However, categorical and non-categorical features are hard to distinguish: perspectives vary about how small the domain size of an attribute needs to be to be regarded as categorical, so we define a numerical vari-

able called "categorical threshold" to help clarify [12]. If the categorical threshold value is 20, attributes with domain sizes less than 20 will be categorical. By contrast, attributes with domain sizes greater than or equal to 20 will be non-categorical.

2.2 Synthetic Data Generation Methods

Before discussing generation models, we introduce the generic concept of synthetic data generation. In general, modeling attribute distributions is an important aspect of successfully generating synthetic data. If the distribution of a real dataset is known beforehand, machine learning methods can be applied to fit the distribution, so similar statistical properties can be maintained in the synthetic data. We define this method as "learning from distribution". As datasets become complex either due to a large domain size or involving a temporal attribute, there will be more combinations of values. Also, the probability of synthesizing some values for one variable can depend on other variables' values, so this introduces conditional probabilities. Modeling this dependency between variables is an important challenge to tackle in synthetic data generation. If it can be captured, then using conditional probability distributions, sampling methods can be applied to produce synthetic data.

2.2.1 Some Synthetic Data Generation Methods

Some common synthetic data generation models are based on deep learning techniques such as variational autoencoders (VAE) and generative adversarial network (GAN). For instance, Miok et al.'s work [10] incorporated VAE with a Monte Carlo Dropout method, lowering the running time while maintaining similar accuracy with baseline methods. In the medical domain, Lu et al. used generative adversarial networks (GAN) [8], and Choi et al. [3] combined autoencoders and generative adversarial network to deal with multi-label discrete variables in patient records. While the

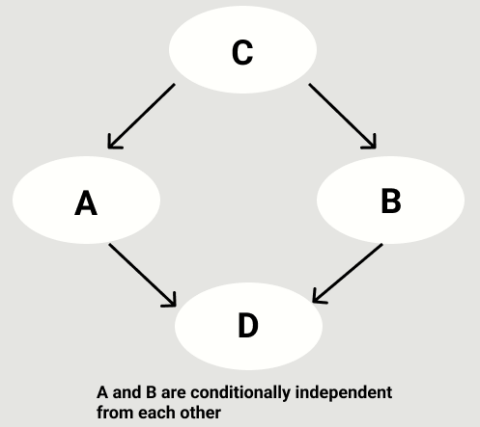


Figure 2.2: A simple Bayesian network: variable C is a root parent, and it has two child attributes, variables A and B. There is no edge between variable A and B which are conditionally independent. Variable D has two parents, so it conditionally depends on both variable A and variable B.

former work confirms the effectiveness of normal GAN models in producing synthetic data, the latter combines the two deep learning tools to address the issue of discrete variables in the medical area.

Another probabilistic model is hidden Markov model. Dahmen and Cook [4] introduced a tool called SynSys to synthesize smart home sensor data, which is based on nested sequences using hidden Markov and regression models. It works well when a limited amount of ground truth data is provided and the data is sequenced by time [4].

Other related works focus on the usability of generation tools. For example, Mannino and Abouzeid developed a tool called Synner, which visually and declaratively specifies properties of the datasets that users intend to generate. Users will also receive instant feedback, so less time is required to complete complex tasks [9].

2.2.2 Bayesian Network

We choose a Bayesian network model to capture probability distributions for creating synthetic results. A Bayesian network uses probabilities to present hidden relation-

ships between different variables and calculate their conditional dependencies. It is a graphical model: the dependency relationships are visualized as edges in a structured directed acyclic graph. When incorporating Bayesian networks in machine learning models, each node represents a corresponded attribute, and the directed edge indicates that a child attribute forms a conditioned distribution depending on its parents' values. One node can have multiple parent nodes, and we call the maximum number of parents the "degree" of a Bayesian network. Bayesian networks assume conditional independence of certain variables, which is an intermediate stage between complete independence and dependence. For example, in Fig 2.2, each node refers to an attribute. If there are two nodes A and B which are not directly connected through an edge, then they are considered to be conditionally independent. That is to say, any extra information of B will not affect the distribution of node A. A challenge for probabilistic models is that it is computationally expensive to capture dependencies among all attributes, so Bayesian networks deal with this issue by using partial conditional independence.

Bayesian networks are also helpful in synthetic data generation because they help capture the probability inferences among attributes in the input dataset, which addresses the conditional dependency problem and helps to determine the sampling order. For example, the distribution of a child attribute will be conditional and sampled after its parents. Thus, Bayesian networks are useful probabilistic model that should serve our goal. There are many works based on Bayesian networks. For instance, Kaur et al. [7] applied Bayesian network techniques to improve existing GAN-based model by dealing with complexity and dimensionality issues in healthcare datasets [7]. Tucker et al. [2] also used Bayesian network to synthesize healthcare data and maintain privacy. Ping et al. [12] developed a user-friendly synthetic generation tool called DataSynthesizer, which is also based on Bayesian network. DataSynthesizer has been applied on urban homelessness data, criminal sentencing data, and some

typical machine learning data [12]. Good results have been shown.

2.3 Evaluation

An effective evaluation method is important to determine the quality of synthetic data. Though there are many evaluation methods in related literature above, some only work for specific datasets. For instance, the smart home dataset in [4] contains location information, so they use Euclidean distance. The goal of [9] focuses on user experience in data generation interface, so usage time is considered an evaluation metric.

There are also some common evaluation metrics, such as utility metrics like correlation matrices and classification accuracy in [3], and statistical features like difference of mean and standard deviation between the matching attributes in real and synthetic datasets [10].

We will leverage both standard and dataset-specific evaluation strategies. First, we will use overall distribution and mutual correlation of synthetic datasets. We will present them through visualizations such as histograms and heat maps. Second, we will evaluate some statistical features that are specific to student course record datasets.

2.4 Comparison to Our Work

None of the related works focused on datasets that contain temporal attributes or attributes with large domain size. Therefore, the performance of those methods for our case is unknown. Similar to the works on smart home sensor data and patient records, model adjustments might be needed for existing generative models to work well for us.

To the best of our knowledge, the only work specific to student data generation

is by Dorodchi et al. [5] who proposed to use the DataSynthesizer, an existing tool developed by Ping et al. [12], to create open access versions of student data. In other words, their goal is to increase data availability. They synthesized a dataset consisting of first-time college students between 2010-2013 and received similar results after training a random forest classifier with both real and synthetic datasets.

While Dorodchi et al.'s work [5] has similar goals, it has two limitations. They only tested a relatively simple dataset with low domain size values and did not evaluate any temporal aspect.

Their work sets the categorical threshold value to be a small number, 20, based on the domain sizes of their categorical attributes [5], and Ping et al. [12] also set it to be 20 for one of their sample datasets. In other words, both [5] and [12] use relatively small threshold value, yet our work targets a more complex dataset with much greater domain size. We intend to explore a method that works well with datasets that have domain sizes around 3000. Student datasets satisfy our goal, since if it has an attribute of course name, this attribute has to be treated as categorical to avoid synthesizing fictional courses. In this scenario, datasets contain categorical attributes with sufficient complexity.

Second, their datasets only record first-time-in-college students [5], which means there is a lack of testing on attribute value changes over time. After a student first enters the university, their information will be collected, but after that, they did not continuously collect data on this specific student. By contrast, our work involves an attribute with temporal nature. Our dataset contains a timestamp feature, so a student is able to appear multiple times, and we intend to capture the change of attributes from the same student in time trend. Some example changes of attribute could be gradually taking higher level courses and earning higher grades.

Chapter 3

Methodology

Our ultimate goal is to identify a synthetic data generation method that works well with complex datasets based on multiple evaluation strategies. First, we will develop a method to synthesize data. The three main steps in this part are acquiring conditional distribution with probabilistic models, maintaining differential privacy [1] (that is, learning from personal or private data without exposing), and sampling based on the distribution results. In particular, we intend to construct a Greedy Bayesian network to extract the value probability of each attribute and store the conditional correlation between two attributes. Then for differential privacy, we inject random noise to the conditional distributions of the Bayesian network. Finally, we will sample from the distribution. Since Bayesian networks also imply sampling orders, there are some well-designed sampling algorithms. An intuitive strategy is starting from the root attribute and moving down in the network following a breadth first approach. Since methods that are based on Bayesian networks are common in the field of synthetic data generation, we first attempt existing tools and explore whether they already provide good results on complex student datasets. When necessary, we will adjust those tools as appropriate.

As previously discussed, our evaluation method includes both standard visualiza-

[1] In this work, we do not evaluate differential privacy.

tion metrics and student-specific statistics.

We tested our dataset on the generation method, evaluated our synthetic output, and iteratively experimented with model configurations to improve performance.

3.1 The DataSynthesizer Tool

We used Dorodchi et al.'s tool [5], DataSynthesizer, since it not only matches our generic approach, but also has been demonstrated to function well for less complex student datasets.

DataSynthesizer inputs a real dataset and generates a synthetic dataset as output. Its distinguishing feature is its usability–while users have the choice to specify parameters manually, the model can infer the information itself. It has three main components: DataDescriber, DataGenerator, and ModelInspector [12]. DataDescriber infers a data summary, which includes both the frequency distribution and attribute information such as data types. DataGenerator generates data in different ways according to the data types. Determined by the categorical threshold value, if an attribute is non-categorical and numerical, DataGenerator uses uniform sampling to sample from an equi-width histogram. For non-categorical string attributes, random string values are generated, and they will have the same string length as real values. By contrast, categorical features will be synthesized by drawing from the frequency distribution obtained from the DataDescriber.

Finally, ModelInspector inspects and compares the overall distribution and mutual correlation between real and synthetic datasets [12]. Side-by-side histograms are used to compare the overall distribution of each category of an attribute, and side-by-side heat maps are presented to compare pairwise mutual correlations.

Moreover, DataSynthesizer can act in three modes: correlated attribute mode, independent mode, and random mode. The correlated attribute mode learns a dif-

ferentially private Bayesian network through a GreedyBayes algorithm to capture the mutual correlation between features. Then, it samples the root attribute from an unconditional distribution and children attributes from conditional distributions, with noises injected. The independent attribute mode draws samples from unconditional distributions and also maintain differential privacy through adding controlled noise. The random mode only generates random values matching the same types of attributes [12].

We did not consider the random model, since completely random data does not meet our goal. Independent attribute mode should be used only when correlated attribute mode is too computationally expensive due to the complexity of Bayesian network or the size of real data is too small to be sufficient in building a reasonable model [12]. For these reasons, we favor the correlated attribute mode. After the mode is chosen, it remains fixed for the experiment.

While the only required input of this tool is a real dataset, there are three essential parameters that can be tuned. The first parameter is the categorical threshold value. The second one is the epsilon value associated with differential privacy, and the general rule is that lower epsilon value leads to higher privacy and lower accuracy. In our work, we will set this value to be maximum 1 to preserve privacy. The final parameter is the degree of Bayesian network, which determines the network's structure. Users also can choose the size of synthetic datasets, but the size is not going to affect the overall accuracy. Thus, this parameter is not a part of our tuning.

Some attributes can be set as the candidate keys of the dataset, which are selected features that act as the key variable. In other words, values of candidate keys are unique, and there is no repetition. This characteristic will also be reflected in synthetic datasets. Candidate key is an optional parameter, and it is fine that users do not assign an attribute to be candidate key.

3.2 Evaluation Method

Our evaluation uses the ModelInspector tool of DataSynthesizer [12] and 11 student-related evaluation statistics. The ModelInspector is utilized to compare overall distribution and mutual correlation, which are two fundamental statistical properties reflecting the quality of synthetic data. With side-by-side histograms and heat maps as visual comparison, this is the most straightforward evaluation strategy.

In addition, we also developed 11 evaluation statistics that work specifically for student course records. Shown in Table 3.1, these features can be roughly divided into two types: temporal and non-temporal types. Non-temporal features are aggregated statistics calculated over the entire time range, including the mean number of courses taken by a student, the mean number of terms that a student takes courses, the percentage of students who take at least one repeated course, the mean and standard deviation of student grades, the mean and standard deviation of student grades grouped by subject, and the mean grade differences between high level and low level courses distinguished by a threshold value. These values can be directly calculated without grouping records by student identities.

By contrast, temporal features require grouping data based on each individual student, including trend analyses on change of students' grades, change of course level, subject changes, and change of the number of course taken. As an example, to explore how students' grade change over terms, we have to first filter out a subset of data that belongs to one student, then calculate grade changes, and finally take the mean of this change over all students. Through this method, we eliminate the effects of factors that some students might take much more courses than others, and some terms that have less data records. Unlike non-temporal features which can be calculated by considering all data together, temporal features require grouping first.

Specifically, as these time analyses are based on terms, the time unit will be one semester. As the capability of Bayesian network to capture temporal correlation is

Non-temporal statistics	
mean-courses:	mean number of courses a student takes
mean-terms:	mean number of terms a student takes
percentage-repeated-courses:	percentage of students who take at least one valid repeated course
mean-grades:	mean of grades
sd-grades:	standard deviation (sd) of grades
mean-sd-subject-grades:	mean and standard deviation of grades grouped by subject
difference-grades:	difference between mean grades of low level & high level courses
Temporal statistics	
change-term-grades:	change of students' grades over terms
change-term-course-level:	change of course level over terms
change-term-count-subjects:	change of count of different subjects over terms
change-term-count-courses:	change of count of number of courses over term

Table 3.1: Student-related evaluation statistics

unknown, we will compare both temporal and non-temporal features between real and synthetic datasets as a quantitative measurement. In case the time attribute of a dataset is not semester based, these statistics could be calculated in a similar way.

Chapter 4

Experiment & Results

4.1 Dataset

Our dataset comprises student course records. The dataset contains the courses taken by all Emory students who have taken at least one Computer Science (CS) track course, like CS or Maths courses. It contains 113894 rows or data records and seven columns or features: student identifier, term, subject, course catalog number, course section, grade, and grade basis. Each record or row represents a student who took a course in a specific term and section and earned either a letter grade or pass/fail depending on the basis.

The student identifier is a random string; term, the only integer type, indicates the semester the course was taken. The term feature covers data from 2008 to 2018. Each term instance is a 4-digit number such as 5129. While the first digit is meaningless, the second and third digits indicate the year this course was taken. The last digit could only be 1, 6, or 9, which respectively means spring, summer, and fall semesters. Thus, 5129 refers to the fall semester in 2012. The rest of the features are string types. For instance, grade is in letter form, and the catalog number sometimes contains one letter, for example, "W" as suffix to indicate the writing nature of a course. In our

work, we plan to treat all features, except student identity which is a random string, to be categorical to avoid unreasonable values, especially course numbers and grades. For instance, we do not want non-existing courses or a mismatched grade and grade basis.

This dataset includes attributes with large domain size and temporal features. First, course information in this dataset has large domain size. The only requirement for a course record to be collected is that it is taken by a student who at least takes one course in CS-track study. That is to say, there are two scenarios. If a student takes an introductory CS-track course but does not pursue further CS study, only the record of this one CS-track course is recorded. Thus, some students only appear once. However, if students continue their CS path, all of the courses, regardless of the subject, taken by this student will be included. Thus, it is reasonable to have subjects like Art or Music in the dataset. This characteristic significantly increases the domain size of subject and catalog number. The domain sizes of the catalog number and subject features are respectively 1383 and 173. To avoid producing fictional courses, it was necessary to consider the subject and catalog number as a fixed pair and regard the pair as a new categorical feature, which resulted in a final domain size of 3711. Hence, we expect our dataset to be sufficiently complex, and our final model must deal with this complexity.

Second, the term attribute of this dataset has a temporal element. Students can take multiple courses throughout different terms, and there is a a total of 17355 unique students within these 113894 records. As the same student identity can be tracked as terms change, we can evaluate if synthetic data generation methods are able to capture student growth and attribute value changes over time.

4.2 Experimental methodology

In this section, we detail our experiment and evaluation.

4.2.1 Preprocessing

Preprocessing can be essential for probabilistic models, since good output can be highly sensitive to the input quality. Our first preprocessing step excluded uninteresting features, like section, and merging highly-connected features, like subject with course catalog number and grade with grade basis. Hence, students will not earn a letter grade with a pass/fail grade basis or take a course that does not exist in Emory University's course curriculum. Meanwhile, we have to treat these two new attributes as categorical for the same reasons.

As a result of our preprocessing, the number of features was reduced from seven to four (student identity, term, grade-basis, and course), but the domain sizes increased due to the combinations. After merging 173 categories of subject and 1383 distinctive course catalog numbers, the domain size of the new course feature is 3711. This new highest domain size will be set as the threshold value, and we claim this merge is acceptable as it does not change the order of magnitude of the domain size.

The second preprocessing step deals with an issue caused by the student identifier attribute in our dataset. Each row of our dataset represents a course taken by a student. Using the student identifier as the candidate key is not appropriate, since it will make each synthetic student identity unique. We do not want random and nonrepeated student identities, since we are interested in whether student growth will be captured by the DataSynthesizer. Thus, generating sets of course records that belong to the same student is important. However, we also do not want to directly treat student identity as categorical, otherwise, we will undesirably select from only a set of students. Our solution was to force repetition by setting a numerical

range less than the size of synthetic dataset, i.e. pigeonholing. To make this work, student identities had to be converted to numerical values, so we mapped each student identity to a random integer and kept a copy of this converted dataset for later experimentation. That is, we now had two preprocessed datasets. One is the normal preprocessed dataset; the other one further converts student identity into numerical form, called the "id-encode preprocessed dataset".

4.2.2 Using DataSynthesizer

We first tested the DataSynthesizer tool on the normal preprocessed dataset due to its relatively lower computational cost. We used the ModelInspetor to check if the overall distribution and mutual correlations are similar to those in the real dataset. When there were obvious differences, we attempted to tune the model parameters to improve the accuracy as described in section 3.1.

When parameter tuning alone did not produce satisfactory synthetic data, we attempted to adjust the overall tool. When modifying DataSynthesizer by adding new functions or algorithms satisfied the ModelInspector evaluation, we then tested with and evaluate the id-encode preprocessed dataset.

4.3 DataSynthesizer with Default Parameters

4.3.1 Experimental Setup

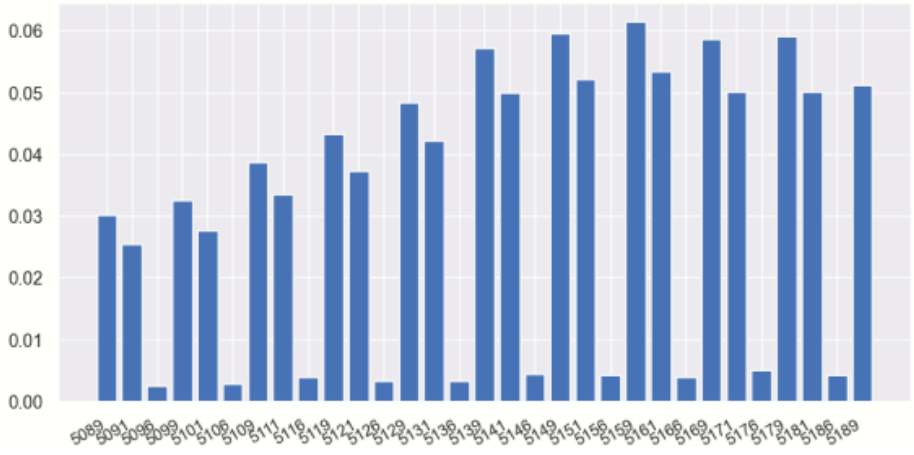
We first tested the normal preprocessed dataset using DataSynthesizer with the correlated attribute mode. We set the categorical threshold to be 3715, slightly more than 3711, which is the greatest domain size among the three categorical features. Epsilon, which controls the noise injection, is set to 1. The degree of the Bayesian network is set to 2, and the size of synthetic dataset is set to 1000. We also set student identity as the candidate key, so it is not considered in evaluation: since

DataSynthesizer will just generate random and unique values for candidate key, there is no need to evaluate it. We ran our experiments based on Ping et al.'s code with version 0.1.8 on https://github.com/DataResponsibly/DataSynthesizer [12].

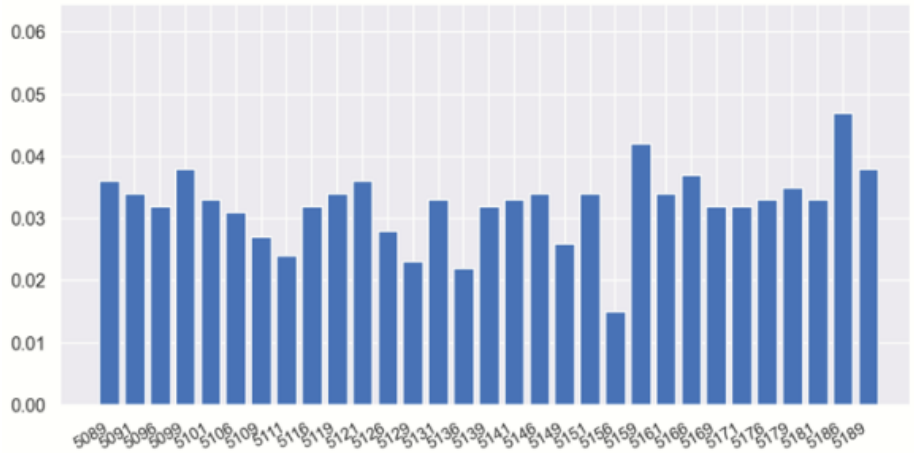
4.3.2 Experimental Results

Figure 4.1 and Figure 4.2 show our results for term and grade-basis. The distribution shapes for the synthetic data generated with the correlated mode are not similar to the real distributions. We also found that the attributes in synthetic data form a distribution that is close to a discrete uniform distribution. The categories that are supposed to occur with much higher frequencies have a lower frequency than those in our real dataset, and vice versa for some categories with low frequency in the real dataset. For instance, in Figure 4.2, approximately 35% of the grades in the real distribution are A's, yet the distribution percentage A's in synthetic data is less than 5%. Similarly, in Figure 4.1, the real distribution maintains two clear patterns. First, every term value ending with 6 will have much lower frequencies than other categories, since those terms refer to summer semesters. Second, in the real datset, there is a general increasing trend from the earliest term to the latest term. However, these two patterns are not captured by synthetic term data for correlated mode. Lastly, all term categories are roughly between 3% and 4 %, which is not similar to the real distribution.

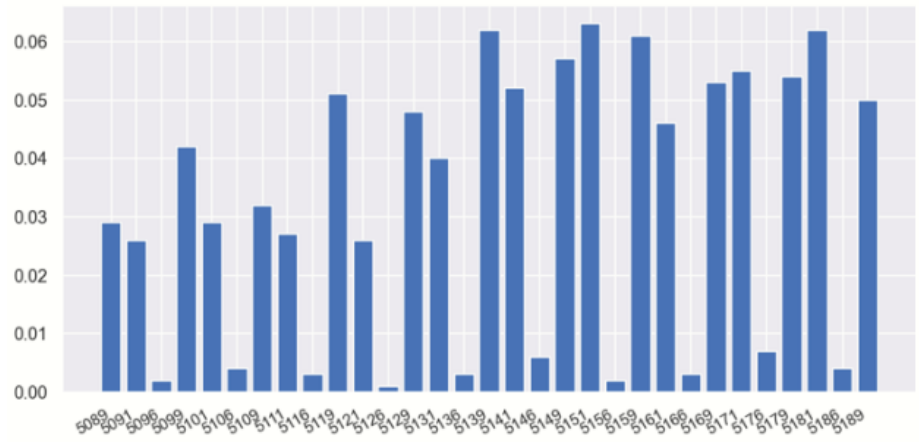
We also tried independent mode and surprisingly found that the synthetic student data generated from the independent mode has a much closer distribution to the real dataset (discussed later). The results are shown in Fig 4.1c and Fig 4.2c. Both figures demonstrate clearly better overall distributions compared to the correlated mode graphs in the middle. This finding is surprising because the correlated attribute mode should generate better synthetic data according to Ping el al.'s work [12]. If independent attribute mode contributes to good results, the correlated attribute mode



(a) Real data.

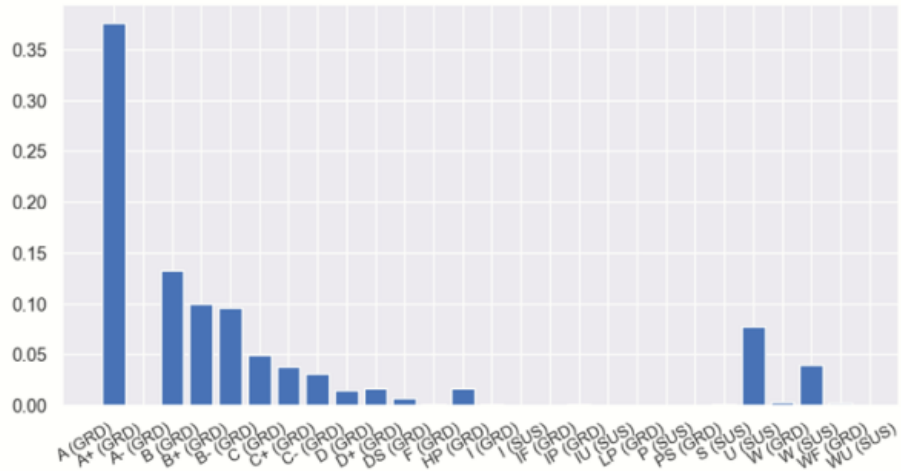


(b) Synthetic data using correlated attribute mode.

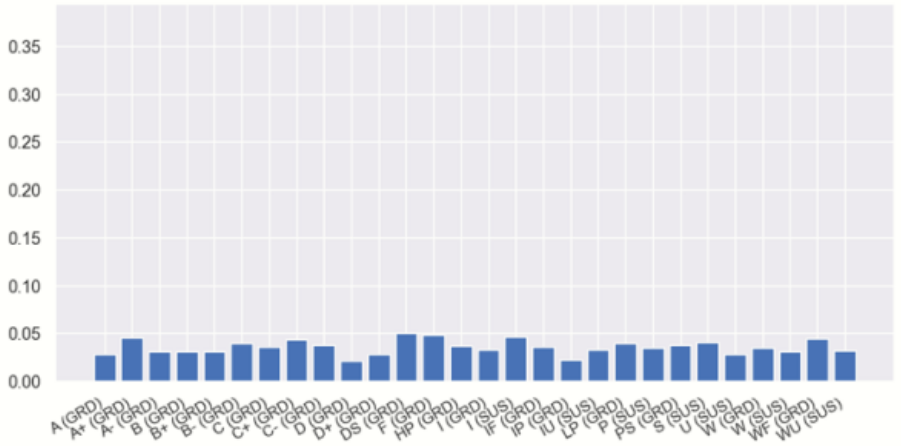


(c) Synthetic data using independent mode.

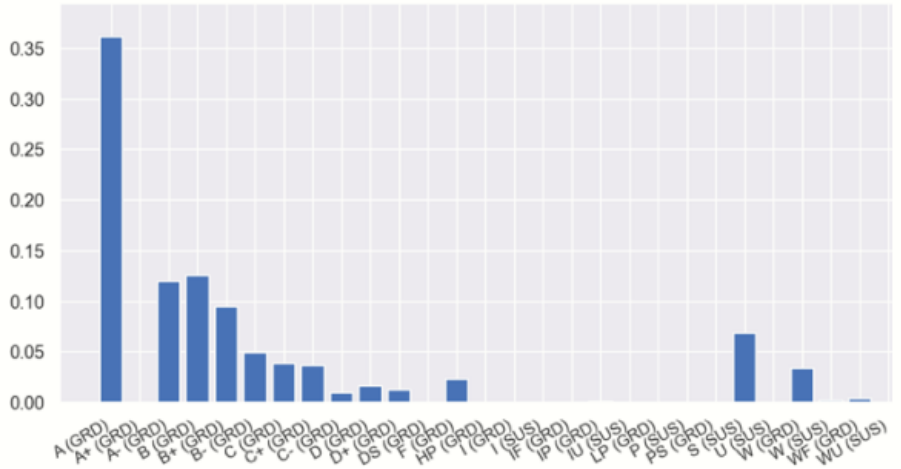
Figure 4.1: Term distributions for DataSynthesizer with default parameters.



(a) Real data.



(b) Synthetic data using correlated attribute mode.



(c) Synthetic data using independent mode.

Figure 4.2: Grade-basis distributions for DataSynthesizer with default parameters.

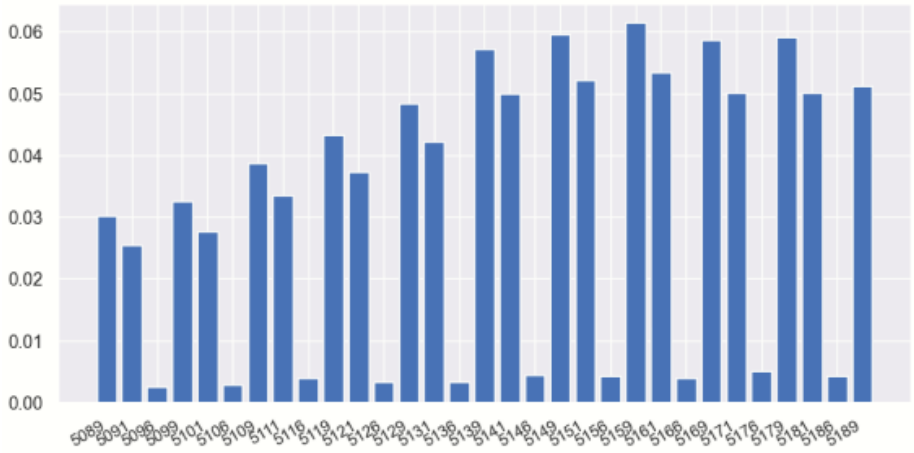
is supposed to be even better.

To tune the correlated attribute mode case, we decreased the max degree of Bayesian network from 2 to 1. Since we only have three categorical features, a relatively small number of attributes, we do not need a more complex Bayesian network that accumulates more noises. The new synthetic results of correlated attribute mode are shown in Figure 4.3c and Figure 4.4c. The comparison indicates small accuracy improvement. However, the results still cannot be considered as good. First, some dominant categories like "A" grades and "A-" grades take roughly 15% and 8% in synthetic data, which are still far from their frequency percentages in the real distribution. Second, some categories that are supposed to take extremely small percentage close to 0% take unusually high percentage in the synthetic data distribution. Consequently, further work was needed to increase synthetic data quality and better understand the correlated attribute mode.

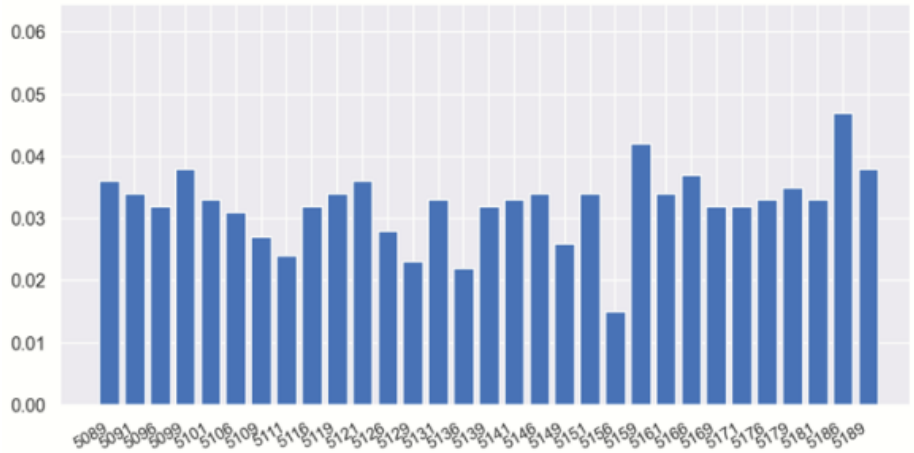
4.3.3 Result Analysis

To figure out the reason for the incorrect distribution with the correlated attribute mode, we checked the correlation coefficients of our dataset, shown in Figure 4.5. Our correlation coefficients are even greater than those used in the sample datasets by the model creators [12], so correlation between attributes should not be the cause.

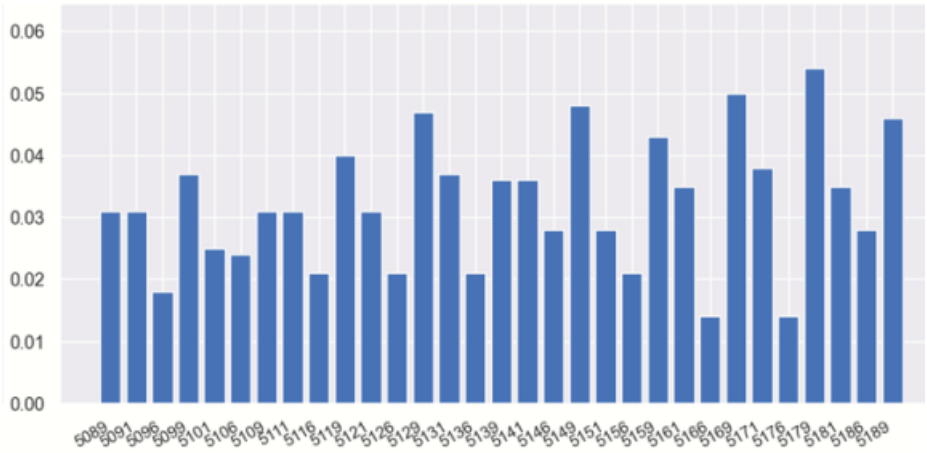
We performed a detailed comparison with the generation process of the sample datasets used by Ping et al. [12]. We found that many distribution-related statistics are actually 10 times less than those in their sample datasets. This explains why parameter tuning helped a little and also why independent attribute mode generates better results. First, having an overly complex network is not necessarily a good thing. Noises added to parents will be passed down to and accumulate in the network, a child attribute with two parents will be injected with more noise than one with only one parent. This extra noise may be too much for our dataset. Since the course



(a) Real data.

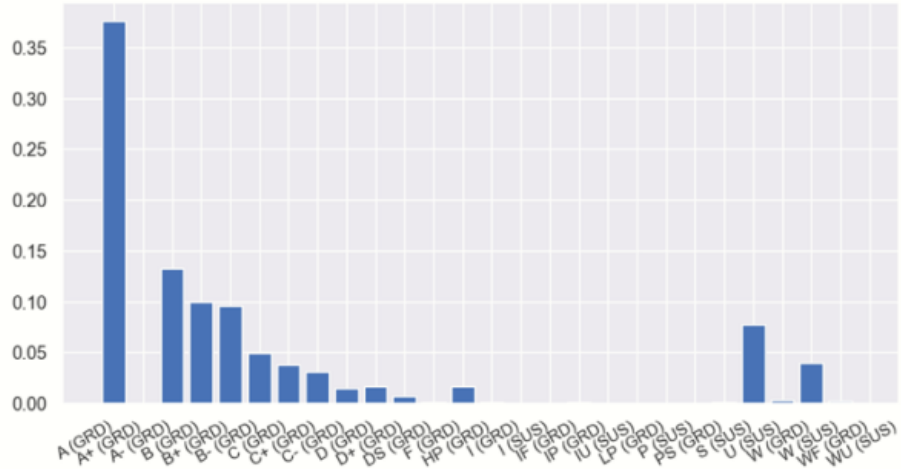


(b) Synthetic data using correlated attribute mode.

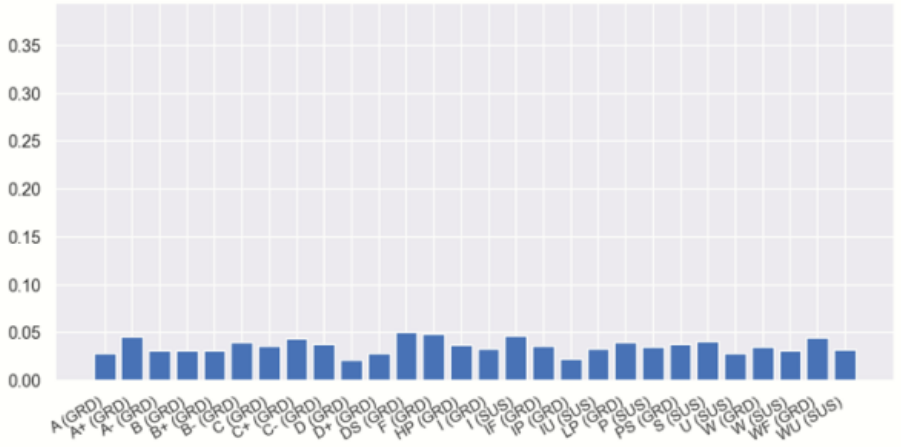


(c) Synthetic data using correlated attribute mode after tuning parameters.

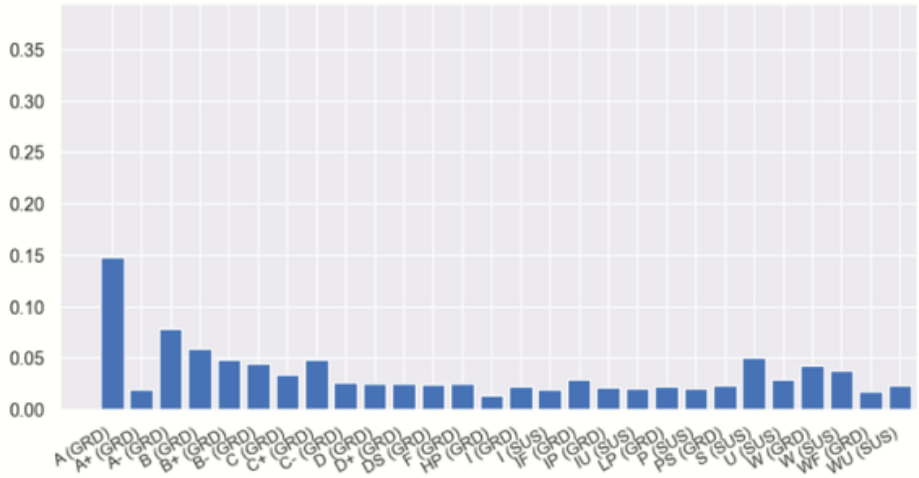
Figure 4.3: Term distributions for DataSynthesizer with default & tuned parameters.



(a) Real data.

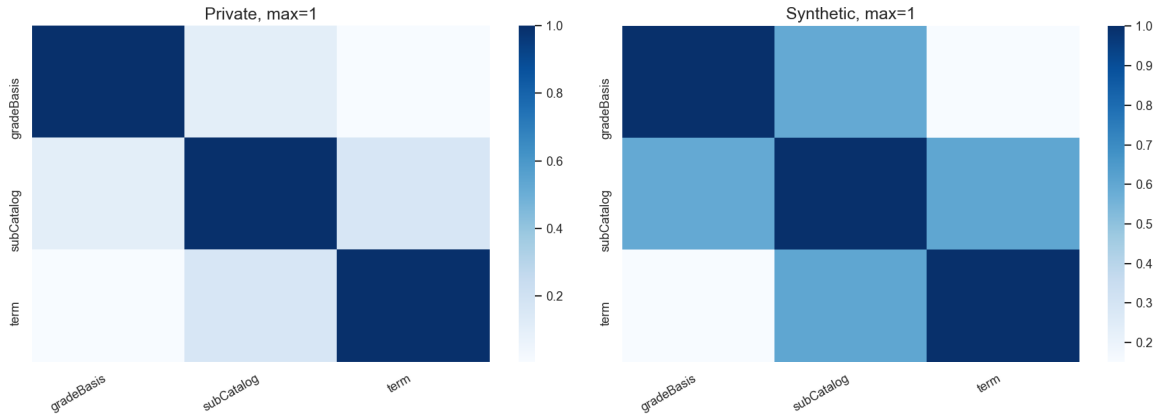


(b) Synthetic data using correlated attribute mode.



(c) Synthetic data using correlated attribute mode after tuning parameters.

Figure 4.4: Grade-basis distributions for tests with default & tuned parameters.



(a) Real data.

(b) Synthetic data using correlated mode.

Figure 4.5: Pairwise mutual correlations for DataSynthesizer test with default parameters.

attribute has a domain size over 3000, conditional probability values will be much smaller compared to those in other related works, so the noise can be too much in a complex network. In Ping et al.'s related work [12], the greatest domain size of their sample datasets is 20, so it is reasonable that their conditional probability values will be greater and suffer less from larger amounts of noise. Thus, by decreasing the degree of Bayesian network, it reduced noise levels in a relatively simple Bayesian network that has only three attributes.

Second, it helps to explain why independent mode reaches a higher accuracy in this case. In particular, independent mode is based on unconditional probability and noise injected into one attribute will not affect others. Noise will follow a Laplace distribution with mean equal to 0 and standard deviation equal to 1. The independent nature of each attribute means that there will not be any noise accumulation. Thus, independent attribute mode will cause less injected noise, and the overall distribution could be better maintained.

4.3.4 Summary

In the first phase of our experimentation, we found that the correlated attribute mode of DataSynthesizer tool does not work well on our complex course record dataset. The main cause is that the domain size of the course attribute in our dataset is too large. Since there are 3711 categories of courses in the dataset, each category will only have a small probability value. These values are too small for the amount of noise injected, resulting into low accuracy. By contrast, parameter tuning and using independent mode increased accuracy, since these two methods reduce the impact of injected noise. Decreasing the number of parents in Bayesian networks passes less noise down to other attributes. In independent mode, there is not even a network, so the noise injected into an attribute will not be affected by other attributes.

In summary, in the first phase, we found that the large domain size of our dataset is a serious problem, and to tackle it through the DataSynthesizer tool, we need to develop a method to decrease domain sizes.

4.4 Incorporating a Split-encode Algorithm

Our previous findings inspired us to develop a method to lower the large domain size. We did this by converting our dataset into another form with a smaller domain size but maintaining all the original information. We now describe this approach and its results.

4.4.1 Split-encode Method & Testing Methodology

We designed a split-encode algorithm to lower the threshold value. The idea is to split the attribute with large domain size into sub-attributes, so the new dataset has more attributes, but each attribute has a smaller domain size.

We encode the course attribute as an integer and split the integer by digits into

separate sub-columns. Since we intended to encode 3711 different courses, we encoded each course into a 4-digit octal number, split the number by digits, and used sub-attributes to store these digits. We converted it to 4 digits, since the total number of combinations of 4-digit octal numbers could cover all 3711 courses in the normal preprocessed dataset.

For instance, if a course is encoded as 1237, then we will drop the course feature and add four new columns that store 1, 2, 3, and 7 respectively. In this case, the four sub-columns, which collectively represents the original course attribute, could only have eight possible values due to the octal number system, a much smaller domain size. As a result, through dividing an attribute with large domain size into several attributes with lower domain size, the threshold value can be decreased.

We then synthesized the dataset with these sub-attributes through the DataSynthesizer tool. After DataSynthesizer produces the synthetic data, we will decode this sub-attributes back to the original attributes. In our case, the only feature to be split is the course, since the domain sizes of grade-basis and term are only around 20, so there is no need to split them.

However, there is also a problem related to using this octal number system: it might synthesize octal numbers are "out-of-scope" and, therefore, cannot be decoded. For example, if the domain size of a course attribute is 3710. The octal number form of 3710 is 7176, which suggests that every course should maintain a one-to-one relationship with numbers from 0 to 7176 in octal number system. However, it is possible to synthesize values from 7177 to 7777, which cannot be decoded back to a valid course value like Math 111. Consequently, our proposed solution is to apply modulo arithmetic. If the domain size is 3710, and the extra octal number is 7177, 7177 will be converted to decimal and then taken the remainder mod 3710. In this case, 7177 is mapped to the course that matches with the modulo. This is reasonable because it avoids decoding fictional courses in an unbiased (uniformly distributed)

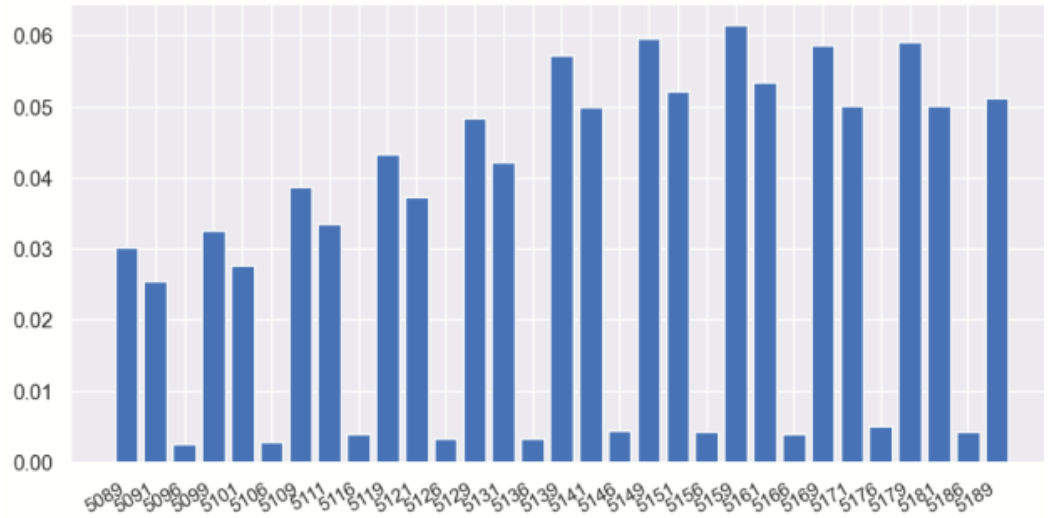
manner. Also, only a few data records will be out-of-scope, and their individual occurrence is small compared to the large size of synthetic datasets: as a result, there should be no significant distribution change.

Our second set of experiments tested whether this split-encode method, through lowering the categorical threshold value, increased synthetic data quality. A concern is whether the Bayesian network will recognize the strong conditional dependencies between the four sub-attributes so that the information of original attributes is not lost. Theoretically, with the Bayesian network's ability to capture correlation between attributes, this should not be the case.

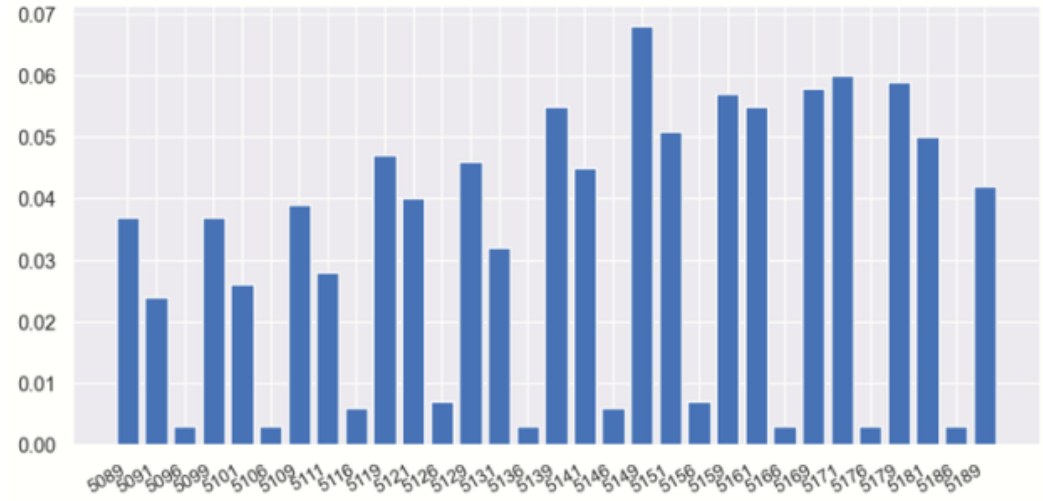
4.4.2 Experimental Setup & Results

After split-encoding the course attribute, the new dataset has seven features: student identifier, term, grade-basis, and the four course digits. We set the epsilon value to 1 and degree of Bayesian network to 2, and the threshold value is set to 32. We followed the same setup in our first experiment and generate 1000 synthetic data. The results are shown by Figure 4.6, Figure 4.7. In Figure 4.6, the synthetic data captures the pattern of summer semesters with small frequency less than 1% and the overall increasing trend. In Figure 4.7, roughly 40% of the grades are A's now, and the distributions of other grades are also close to the real ones.

The comparative result of the course attribute is shown in Figure 4.8 after decoding the 4 sub-attributes back to the original form. It is not shown as visual graph, since it is hard to visualize details given this large amount of categories, so we only compared the distribution frequencies of the top 7 courses with most occurrences. The result is very good, as the frequency percentages of most courses in the synthetic dataset are only different from real results by approximately 0.3%, and the greatest percentage difference is still less than 1%.

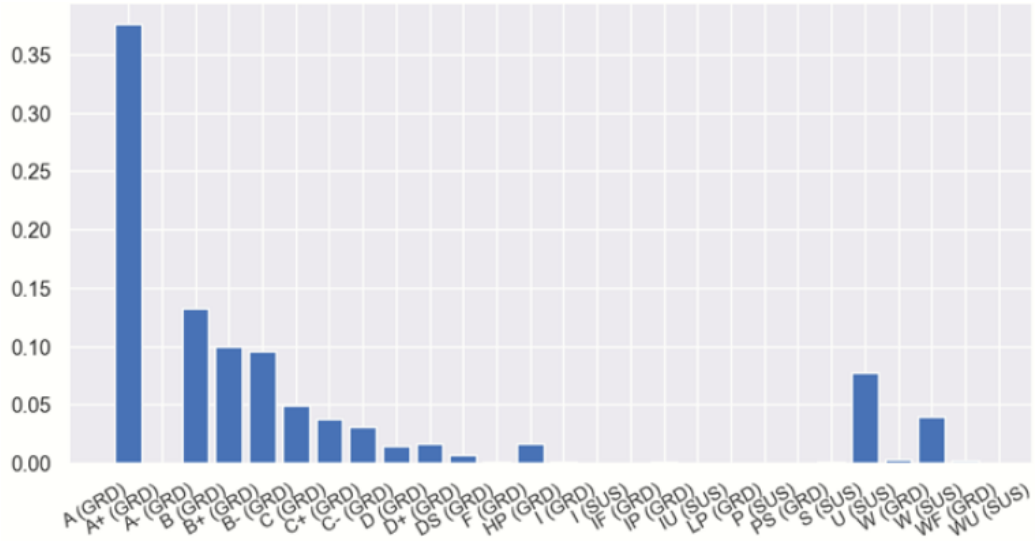


(a) Real data.

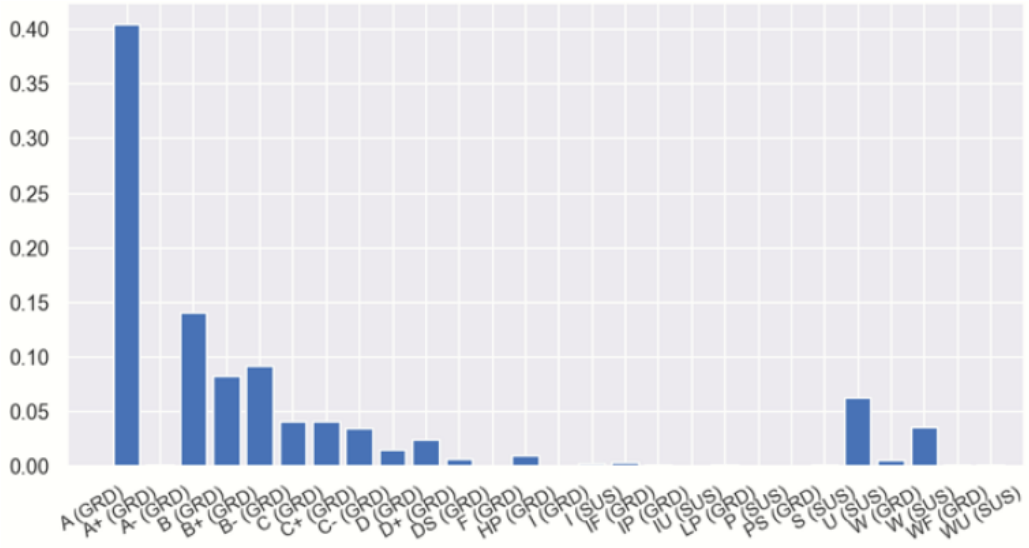


(b) Synthetic data using correlated attribute mode.

Figure 4.6: Term distributions for DataSynthesizer after split-encoding the input.



(a) Real data.



(b) Synthetic data using correlated attribute mode.

Figure 4.7: Grade-basis distributions for DataSynthesizer after split-encode the input.

subCatalog	frequency
MATH 111	0.053
MATH 112	0.029
CS 170	0.027
MATH 211	0.024
MATH 221	0.022
MATH_OX 111	0.015
MATH 212	0.015

(a) Real data.

subCatalog	frequency
MATH 111	0.045
MATH 112	0.026
CS 170	0.024
MATH 211	0.023
MATH 221	0.022
MATH_OX 111	0.017
MATH 212	0.017

(b) Synthetic data using correlated mode.

Figure 4.8: Distributions of the top seven courses with most occurrences after after incorporating a split-encode algorithm.

4.4.3 Result Analysis

It appears clear that for datasets with large domain size, if we can manage to lower the categorical threshold value, which requires decreasing the domain size of some complex attributes, the overall quality could be significantly improved. In summary, what we have achieved is an almost perfect match of overall distribution for attributes with small domain size, grade-basis and term in this case, and a decent synthetic data distribution of attributes with high domain size such as course.

4.4.4 Summary

In this phase, we tested a split-encode algorithm to reduce data complexity, and the results were good. We think this addresses the large domain size issue.

The main goal of the first two phases are testing the performance of DataSynthesizer tool on datasets with large domain size. Now remains our test of the dataset that encodes student identifier in numerical form.

4.5 DataSynthesizer with ID-encoded Data

Encoding student identifiers is important, since it addresses the issue of non-repeating student identifiers in the synthetic dataset. If all student identities are unique, some of our evaluation statistics become impossible to calculate. For instance, if we have no idea which subset of courses are taken by the same student, there is no way to calculate any temporal statistics per student.

If student identity is in numerical form, we can add a numerical range restriction within the DataSynthesizer. For example, the id-encode dataset has 17353 different student identities within 113872 data records. If the synthetic dataset size is set to be the same as the original dataset, and the numerical range is set to be from 0 to 17353, there will be lots of repetitions of student identities. Besides, student identity will be treated as a non-categorical integer type in this case, which means that this process will not affect categorical threshold and data quality. However, since it is non-categorical, it also indicates that the distribution of student identity will be compared for a range of values instead of individual value. Unlike categorical attributes where the distribution frequency is considered for each value, there is no concept of category for student identity, so it will not strictly follow the real distribution. The average amount of occurrences for each student is 7 for both the real and synthetic dataset. However, their distributions are different. For example, in the real dataset, 45% and 23% of students appear once and twice respectively. For all amount of identifier occurrences that are greater than 6, each frequency percentage is less than 1%. By contrast, the distribution of student identifiers in the synthetic dataset is relatively uniform. 16%, 15%, 15%, and 12% of students appear 6, 5, 7, 8 times respectively. Students who appear once take 0.8%. This distribution discrepancy is unavoidable, since student identifier cannot be regarded as categorical.

4.5.1 Experimental Methodology & Setup

After this encoding process, student identity will not be considered as candidate key, and it will be added as a new attribute in evaluation. Our final setup makes threshold value to be 32 and sets the epsilon to 1 and degree of Bayesian network to 2. We also added a numerical range restriction on student identifier from 0 to 17353, and the size of the synthetic dataset was changed to 113872, same as the id-encode dataset.

4.5.2 Results & Analysis

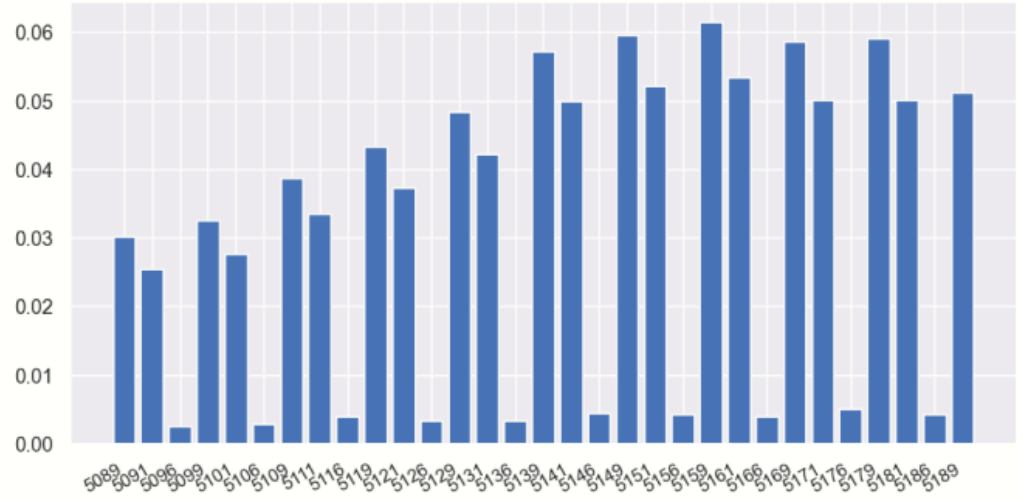
The results are shown in Figure 4.9, and Figure 4.10. Similar to the second phase experiment, all attributes have good frequency distribution results. Once again, the evaluation of the course attribute, shown in Figure 4.11, is done separately due to its large domain size.

We confirmed that the overall distribution quality is not harmed by our identity encoding strategy, which allows us to compute all relevant statistics.

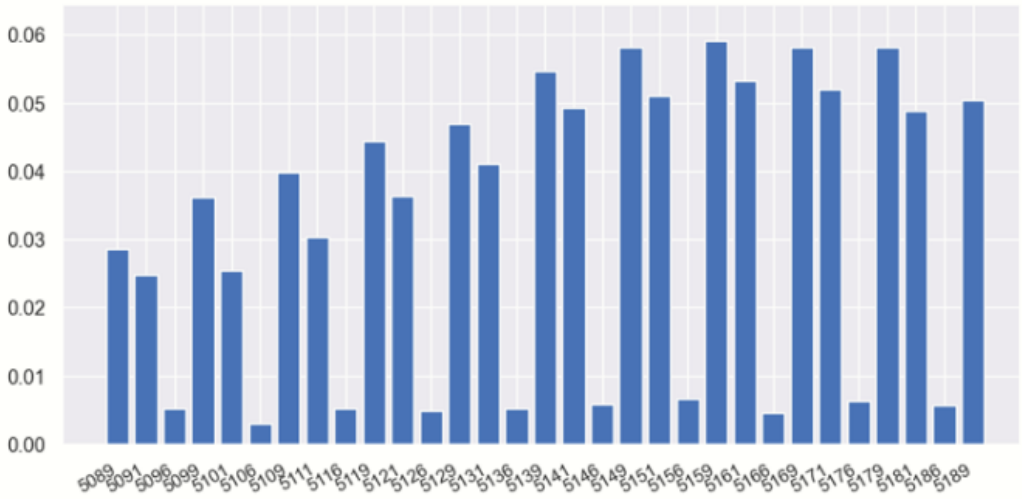
The term and grade-basis attributes maintain similar accuracy as the second phase experiment, and the course results are slightly better than the previous results. The distribution frequencies of all of the top 7 courses are close to real values, similar to the results in the second phase.

4.6 Evaluation statistics

Some of our student-specific evaluation statistics are related to student's GPA, such as the mean and standard deviation of GPA for overall students or the trend analysis on students' grade. Thus, we created a copy of our dataset, which converts letter grades in the real dataset to corresponded numeric grade points. We only kept the normal letter grades from A to F and removed the grade such as satisfied, unsatisfied or incomplete, since only letter grades are counted as part of GPA. Also, some spe-

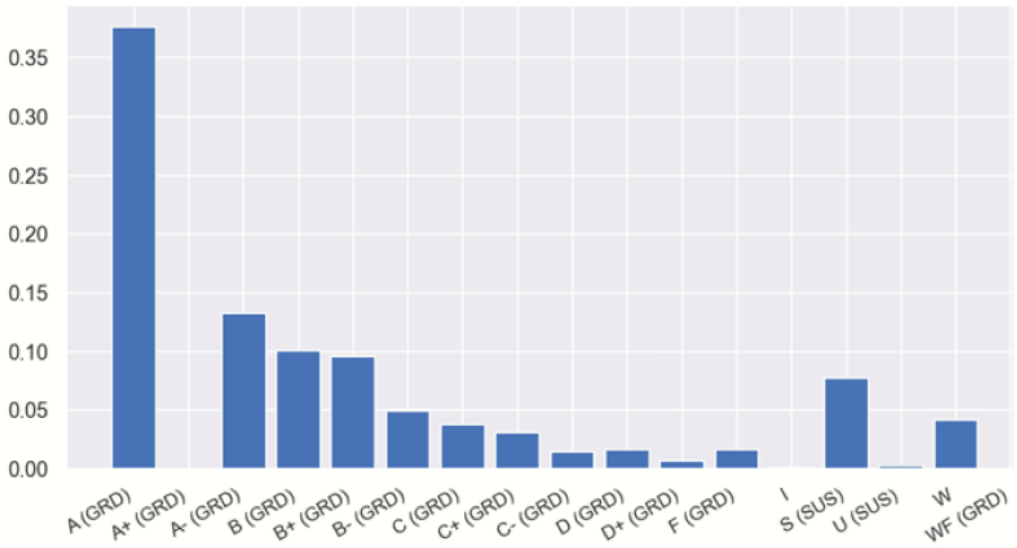


(a) Real data.

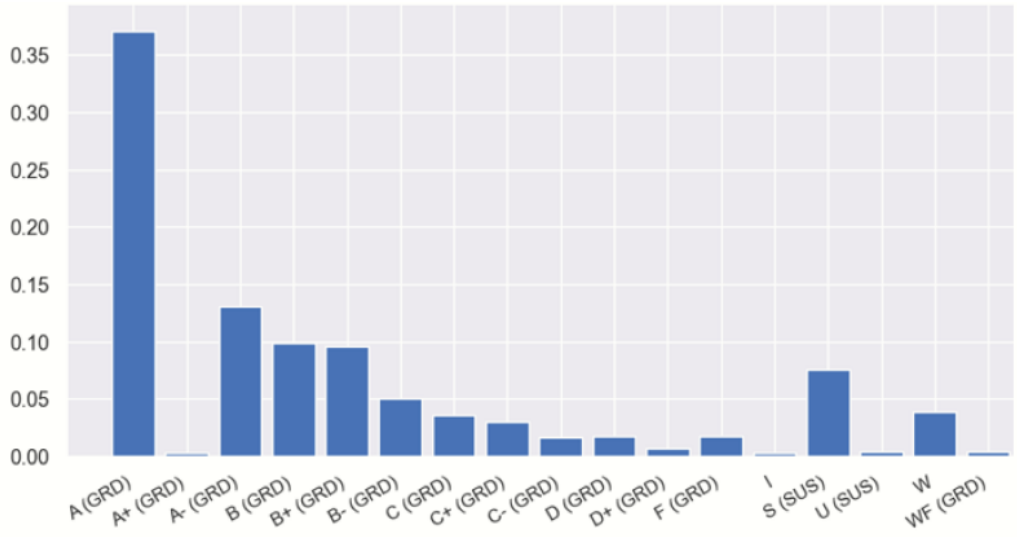


(b) Synthetic data using correlated attribute mode.

Figure 4.9: Term attribute distributions for DataSynthesizer test after incorporating the id-encode strategy.



(a) Real data.



(b) Synthetic data using correlated attribute mode.

Figure 4.10: Grade-basis attribute distributions for DataSynthesizer test after incorporating the id-encode strategy.

subCatalog	frequency
MATH 111	0.053
MATH 112	0.029
CS 170	0.027
MATH 211	0.024
MATH 221	0.022
MATH_OX 111	0.015
MATH 212	0.015

(a) Real data.

subCatalog	frequency
MATH 111	0.045
MATH 112	0.026
CS 170	0.024
MATH 211	0.023
MATH 221	0.021
MATH_OX 111	0.017
MATH 212	0.016

(b) Synthetic data using correlated mode.

Figure 4.11: Distributions of the top seven courses with most occurrences after after incorporating the id-encode strategy.

cial grades such as those in Emory Goizueta Business School will also be converted accordingly to numerical values. This copy of dataset is saved for evaluation.

Then the evaluation statistics are calculated for both the synthetic and original datasets, and the result is shown in Table 4.1. Repeated courses are special courses that have course values ending up with a "R" in Emory course curriculum. Normally, students are not allowed to take the same course, but repeated courses are exceptions, so the percentage of students who take at least one valid repeated course is also considered as part of the evaluation.

A general student course trend is reflected by gradually taking higher level courses, so we compared the difference between average grade of low level and high level courses. The level is distinguished by a threshold value, and in this case we regard catalog number less than 300 as low level and greater than or equal to 300 as high level. This threshold value is subjective.

The rest of the statistics are temporal statistics. They reflect the overall trend change of a particular interesting statistics for all students. We grouped data by student identifier, calculated the relevant statistics for each student over each term, sorted the results based on term in ascending order, calculated the average of value

changes between each term, and finally average this result over all students. For instance, the change-term-grades is calculated through the formula below.

$$change{-}term{-}grades = \text{Avg of [Avg difference of successive term GPA] for overall student} \quad (4.1)$$

As for those temporal statistics, grade change and course level change demonstrate how the student's grade and course level change over time. The count of different subjects indicates if a student will gradually move from studying a range of subjects to focusing only on major subjects. Finally, count of number of courses is related to another pattern. Some students prefer taking less classes in beginning terms, which causes heavy workload at the end. Some students do the opposite thing. Thus, this is another way to tell if a student dataset is real.

Most of the non-temporal statistics have similar values. The value difference of mean-courses, percentage-repeated-course, sd-grades, and mean-grades are all less than 0.1. The synthetic result of difference-grades is off by 0.11, which is still acceptable. The values of all temporal statistics are close to 0 with an error less than 0.1. This is a good result that suggests that our grade-specific attributes are generated effectively. There is a relatively big difference for mean-terms, different by 3 terms. This is likely caused by our method that treats student identifier as non-categorical. This statistics require grouping by student identifier first, yet the generation method does not strictly synthesize the real student distribution. This also explains why we deem 0.1 as an acceptable error range for temporal statistics.

statistics	real data results	synthetic results
mean-courses	7 courses	7 courses
mean-terms	3 terms	6 terms
percentage-repeated-courses	12.95%	13.47%
mean-grades	mean:3.36	mean:3.35
sd-grades	sd: 0.86	sd:0.87
mean-sd-subject-grades	shown separately	shown separately
difference-grades	0.17	0.06
change-term-grades	-0.11	0.00
change-term-course-level	0.12	0.04
change-term-count-subjects	-0.06	0.01
change-term-count-courses	0.00	0.01

Table 4.1: Student-related evaluation statistics results for real and synthetic datasets. The 6th statistics is shown in Appendix A due to its large size.

Chapter 5

Conclusion & Future Work

In this thesis, our research goal was to identify a synthetic data generation method that produces high quality data for complex datasets with attributes that have large category size. To achieve this goal, we studied a complex student dataset. We evaluated an existing tool, called DataSynthesizer, using both standard visualizations and student-specific statistics. We conclude that the DataSynthesizer tool, with the aid of our new split-encode algorithm, can generate quality synthetic datasets with complex features.

While this work shows a method for generating quality data, there remains several opportunities for further explorations, including:

- exploring other data generation models that may be able to intrinsically deal with complex datsets like ours. These models may include variational autoencoders (VAE) and generative adversarial network (GAN).

- deeper analysis of the effects of model parameters, like epsilon, choice of the root attribute, and the degree of Bayesian networks. They all affect the quality of synthetic results. Epsilon value is inversely related to the amount of noise injected. Changing the root attribute or the degree affects the structure of the Bayesian network.

- exploring the acceptable amount of overlapping data records. Synthetic datasets should be similar to the real dataset, so it is normal to have identical records (except the identifier attribute) between the real and synthetic datasets. However, exploring how much overlapping data are tolerable is another future work.

- evaluating privacy metrics. Differential privacy prevents information leakage of sensitive data. Though it is claimed to be preserved in the DataSynthesizer tool, we have not yet evaluated this claim.

- using machine learning tasks to evaluate the quality of synthetic data. We can further validate the quality of our synthetic data, for example, if two models, trained by the real and synthetic dataset respectively, have similar accuracy.

- testing the same generation methods on a broader range of student or other complex datasets.

Appendix A

More Experimental Details

First 20 subjects sorted by frequencies in the real dataset		
subjects	real data results	synthetic results
MATH	count:29628 mean:3.20 sd:0.95	count:27825 mean:3.29 sd:0.90
CS	count:11060 mean:3.21 sd:0.99	count:10426 mean:3.30 sd:0.90
MATH_OX	count:7739 mean:3.00 sd:1.09	count:7585 mean:3.23 sd:0.98
ECON	count:6639 mean:3.44 sd:0.73	count:6114 mean:3.47 sd:0.75
BUS	count:4948 mean:3.42 sd:0.59	count:5490 mean:3.40 sd:0.81
PE	count:3030 mean:3.84 sd:0.54	count:2969 mean:3.41 sd:0.84
PHYS	count:3013 mean:3.39 sd:0.76	count:2895 mean:3.39 sd:0.84
CHEM	count:2721 mean:3.28 sd:0.81	count:2535 mean:3.41 sd:0.80
BIOL	count:1889 mean:3.41 sd:0.73	count:1614 mean:3.48 sd:0.77
ENG	count:1837 mean:3.61 sd:0.62	count:1662 mean:3.51 sd:0.71
MUS	count:1535 mean:3.87 sd:0.45	count:1411 mean:3.49 sd:0.81
PHIL	count:1046 mean:3.64 sd:0.59	count:1027 mean:3.40 sd:0.84
HLTH	count:1000 mean:3.75 sd:0.48	count:1178 mean:3.43 sd:0.85
PE_OX	count:961 mean:3.84 sd:0.51	count:880 mean:3.36 sd:0.85
SPAN	count:884 mean:3.62 sd:0.55	count:710 mean:3.44 sd:0.79
POLS	count:863 mean:3.43 sd:0.74	count:819 mean:3.35 sd:0.90
PSYC	count:818 mean:3.27 sd:0.84	count:756 mean:3.36 sd:0.85
JPN	count:774 mean:3.56 sd:0.74	count:763 mean:3.45 sd:0.77
CHN	count:759 mean:3.65 sd:0.56	count:712 mean:3.40 sd:0.85
ECON_OX	count:722 mean:3.44 sd:0.66	count:640 mean:3.30 sd:0.84

Table A.1: Statistics of count, mean, and standard deviation statistics of the first 20 subjects that are sorted by frequencies. While all subjects are recorded, it is hard to put them in one table, so part of them are shown as examples.

Grades sorted by frequencies in the real dataset		
grades	real data results	synthetic results
A	count:42839	count:41898
A-	count:15117	count:14735
B	count:11462	count:11421
B+	count:10883	count:10618
S	count:8863	count:8759
B-	count:5632	count:5802
W	count:4698	count:4825
C	count:4278	count:4460
C+	count:3485	count:3430
D	count:1859	count:2036
F	count:1826	count:1720
C-	count:1698	count:1673
D+	count:801	count:1081
U	count:321	count:353
I	count:81	count:246
WF	count:20	count:279
A+	count:8	count:232

Table A.2: Statistics of count of each grade in our dataset, sorted by grade frequencies in the real dataset.

Bibliography

[1] George Adam, Ladislav Rampášek, Zhaleh Safikhani, Petr Smirnov, Benjamin Haibe-Kains, and Anna Goldenberg. Machine learning approaches to drug response prediction: challenges and recent progress. *NPJ Precision Oncology*, 4, 2020.

[2] Juan Benedetti, Namir Oues, Zhenchen Wang, Puja Myles, and Allan Tucker. *Practical Lessons from Generating Synthetic Healthcare Data with Bayesian Networks*, pages 38–47. 01 2020. ISBN 978-3-030-65964-6. doi: 10.1007/978-3-030-65965-3_3.

[3] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks, 2018.

[4] Jessamyn Dahmen and Diane Cook. Synsys: A synthetic data generation system for healthcare applications. *Sensors*, 19:1181, 03 2019. doi: 10.3390/s19051181.

[5] Mohsen Dorodchi, Erfan Al-Hossami, Aileen Benedict, and Elise Demeter. Using synthetic data generators to promote open science in higher education learning analytics. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4672–4675, 2019. doi: 10.1109/BigData47090.2019.9006475.

[6] Andre R Goncalves, Priyadip Ray, Braden Soper, Madhumita Myneni, Jennifer L

Stevens, Linda M Coyle, and Ana Paula Sales. Generation and evaluation of medical synthetic data, 2019.

[7] Dhamanpreet Kaur, Matthew Sobiesk, Shubham Patil, Jin Liu, Puran Bhagat, Amar Gupta, and Natasha Markuzon. Application of Bayesian networks to generate synthetic health data. *Journal of the American Medical Informatics Association*, 28(4):801–811, 12 2020. ISSN 1527-974X. doi: 10.1093/jamia/ocaa303. URL `https://doi.org/10.1093/jamia/ocaa303`.

[8] Pei-Hsuan Lu, Pang-Chieh Wang, and Chia-Mu Yu. Empirical evaluation on synthetic data generation with generative adversarial network. pages 1–6, 06 2019. ISBN 978-1-4503-6190-3. doi: 10.1145/3326467.3326474.

[9] Miro Mannino and Azza Abouzeid. Synner: Generating realistic synthetic data. *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020.

[10] Kristian Miok, Dong Nguyen-Doan, Daniela Zaharie, and Marko Robnik-Šikonja. Generating data using monte carlo dropout. In *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 509–515, 2019. doi: 10.1109/ICCP48234.2019.8959787.

[11] Omar Mohamed Salim, Hassen Taher Dorrah, and Mahmoud Adel El-Kahawy. A novel algorithm to generate synthetic data for continuous-state stationary stochastic process (wind data application). In *2018 Twentieth International Middle East Power Systems Conference (MEPCON)*, pages 333–338, 2018. doi: 10.1109/MEPCON.2018.8635270.

[12] Haoyue Ping, Julia Stoyanovich, and Bill Howe. Datasynthesizer: Privacy-preserving synthetic datasets. pages 1–5, 06 2017. doi: 10.1145/3085504.3091117.

[13] Gabriel Villalonga, Joost Van de Weijer, and Antonio M. López. Recognizing new classes with synthetic data in the loop: Application to traffic sign recognition. *Sensors*, 20(3), 2020. ISSN 1424-8220. doi: 10.3390/s20030583. URL `https://www.mdpi.com/1424-8220/20/3/583`.