On Defining Rules for Cancer Data Fabrication*

Juliana K. F. Bowles[1][0000−0002−5918−9114], Agastya Silvina[1][0000−0002−0012−9256], Eyal Bin[2], and Michael Vinov[2]

[1] School of Computer Science, University of St Andrews
St Andrews KY16 9SX, UK
{jkfb|as362}@st-andrews.ac.uk
[2] IBM Research Laboratory, Haifa, Israel
{bin|vinov}@il.ibm.com

Abstract. Data is essential for machine learning projects, and data accuracy is crucial for being able to trust the results obtained from the associated machine learning models. Previously, we have developed machine learning models for predicting the treatment outcome for breast cancer patients that have undergone chemotherapy, and developed a monitoring system for their treatment timeline showing interactively the options and associated predictions. Available cancer datasets, such as the one used earlier, are often too small to obtain significant results, and make it difficult to explore ways to improve the predictive capability of the models further. In this paper, we explore an alternative to enhance our datasets through synthetic data generation. From our original dataset, we extract rules to generate fabricated data that capture the different characteristics inherent in the dataset. Additional rules can be used to capture general medical knowledge. We show how to formulate rules for our cancer treatment data, and use the IBM solver to obtain a corresponding synthetic dataset. We discuss challenges for future work.

Keywords: Cancer data · Synthetic Data · Constraint Solvers · Fabrication Rules

1 Introduction

Data accuracy is crucial for being able to trust the results obtained from any machine learning models. Previously, we have developed machine learning models for predicting the treatment outcome for breast cancer patients that have undergone chemotherapy at a health board in Scotland [11], and developed a monitoring system for their treatment timeline showing interactively the options and associated predictions [12]. Available cancer datasets, such as the one used in our work, are often too small to obtain significant results, and make it difficult to explore ways to improve the predictive capability of the models further. Within the options available with machine learning and deep learning,

* This research is partially funded by the Data Lab, and the EU H2020 project Serums: Securing Medical Data in Smart Patient-Centric Healthcare Systems (grant 826278).

we often require substantially more data than we can get access to. Even though we have direct access to the oncology dataset within the local health board, it is not easy to extract the required quantity of data for developing our model. Indeed, there may not be enough data available to perform a suitable analysis.

We explore an alternative approach to enhance our cancer dataset through synthetic data generation. This approach gives us enough data to design proof-of-concept enhanced prediction models. From our original dataset, we extract rules to fabricate data. These rules must formulate exactly the characteristics of the original dataset. Further rules can be added to capture general medical knowledge and information that a small dataset may not contain. This paper shows how to formulate all required rules for our cancer treatment data, which will enable us to obtain a corresponding synthetic dataset. An added complexity in our dataset is the relationship between different events throughout the treatment of a patient. Hence, to generate realistic synthetic datasets, we have to be able to capture accurately the various constraints associated to a treatment as well as possible relationships between events. We will show that the IBM Data Fabrication Platform allows us to capture these complex constraints as needed.

This paper is structured as follows: Section 2 motivates our approach, presents related work, and describes the structure of the original dataset and some restrictions on what is involved in a chemotherapy treatment for a given patient. Section 3 gives a brief description of the IBM data fabrication platform, and shows how to obtain the rules for our cancer treatment dataset. We conclude in Section 4 with a discussion of future work.

2 Motivation, Related Work and Cancer Data

Obtaining accurate toxicity prediction models in cancer care is vital, as it can help identify treatments that are not suited to a patient, and thus improve their outcome overall. However, cancer treatment data, and healthcare data in general, may be limited or difficult to access due to its sensitive and private nature.

There are advantages of using synthetic data in the healthcare domain. Fabricated data allows us to start building models without the need to access real data. We can fabricate large-scale datasets quickly, which allows us to improve the model to resist over-fitting (often a problem with small datasets). Furthermore, the use of synthetic data enables us to simulate outlier events (e.g., rare diseases). Note that we later need to retrain the model with the real dataset.

One option to generate fabricated data involves the use of an existing dataset and imputing the values for a desired field. Rubin [10] proposed the idea of using multiple imputations for all the data-points in the dataset to generate a (partial or complete) synthetic dataset. In statistics, imputation is the process of replacing missing data with substituted values. Given enough data and iterations, it is possible to generate a synthetic dataset for specific purposes. With the rise of machine learning in data mining, Reiter et al. [8, 4] extend the idea of using multiple imputers by using several machine learning algorithms to generate synthetic data. There are many data synthesisers [5] available with machine learning models (e.g., linear regression, random forest, decision tree, neural networks) in their backbone. However, machine learning is not well suited for this task: we need sufficient data to be able to infer a pattern in the dataset, and machine learning cannot capture data sequences accurately. Data sequences arise in a cancer dataset, where every entry corresponds to a patient event (e.g., hospital visit, treatment), and several events form an ordered sequence in the treatment.

Generating data with potentially complicated dependencies, requires the use of solvers, such as *constraint satisfaction problem* (CSP) solvers [13] or *satisfiability modulo theories* (SMT) solvers [7]. As an example, there is a solution that generates data for form-centric applications using an SMT solver [2]. Although this may avoid many of the challenges of other data formats, such as relational databases with complex topologies and hierarchical structures, the solution uses workarounds which can introduce an under approximation of the solution space, thus yielding additional complications to the solver. These complications affect the performance and the scalability of the technology, as well as the quality of its results. In our approach we use the IBM solver which avoids these issues.

We use a cancer dataset extract from a Scottish health board from 2014 to 2016, consisting of Scottish Morbidity Records[3] which includes *SMR01* (hospital admission data), *SMR06* (cancer registry), and *Charlson Comorbidity Index* (categorising the coexistence of a chronic condition with cancer [9]); National Records of Scotland (e.g., *Data on Deaths*); the Oncology DCO database which includes *Demographics* (e.g., date of birth, gender, ethnicity), *Diagnosis* (e.g., cancer stage and site), *Surgery* and *ChemoCare* (e.g., chemocare_general and chemocare_toxicity). In addition, note that a patient in Scotland is uniquely identified by a *Community Health Index* (CHI).

A cancer patient may be given a series of different treatments, known as a treatment pathway. New patients undergo different sets of tests (e.g., MRI, CT SCAN) to determine the type of cancer and the first treatment to be given. There are several types of primary and follow-up treatments, but we focus on chemotherapy treatments here. Chemotherapy uses one or more anti-cancer drugs as part of a standardised chemotherapy regimen, and may be given with a curative intent, or with a palliative intent where the aim is to prolong life or to reduce symptoms. Overall the treatment is very aggressive and it affects the toxicity levels of the patient, particularly in case of comorbidities. Predicting toxicity levels is thus important throughout the treatment in order to be able to adjust it for the wellbeing of the patient. The general pattern of chemotherapy, important to define correct rules for data fabrication, is given below:

- A patient can only be treated with one intention or purpose of the treatment, such as, curative, palliative, adjuvant (an add-on therapy).
- After a specific time has passed, in case of cancer relapse, the patient might be given another treatment with a different intention.
- Each intention has several different regimens.
- Each regimen has several different drugs.

[3] See https://www.ndc.scot.nhs.uk/National-Datasets/ for Information on SMR datasets.

– The treatment may last for several weeks or months that is given in cycles. Hence, each regimen may have more than one cycle.
– A patient may be given several regimens at a time.
– Some regimens may belong to one protocol.

3 The IBM Data Fabrication Platform and Cancer Rules

We use the IBM Data Fabrication Platform (DFP) to generate synthetic data for our application [6]. DFP is based on rule-guided fabrication whereby the data and metadata logic is extracted from the underlying real data or its description and is modelled using rules that the platform provides. DFP allows for new rule types to be added by users. Once a user requests the generation of a certain amount of data into a set of test databases or test files, the platform ensures that the generated data satisfies the modelled rules as well as the data consistency requirements. The platform is capable of generating data from scratch which we do for our dataset. We define the rules, type of data, volume of data, and the relationships among different columns in the dataset. The rule types include:

– Constraints: domains, mathematical functions, arithmetical relations, string relations, regular expressions.
– Knowledge: chosen from existing data sources.
– Analytics: value and pattern distributions, smart classifications.
– Transformations: constraints describing relations between targets and sources, can be bundled to transform tuples.
– Programmatic rules: user-defined code/script functions that generate target values.

Once the user has defined the data sources and rules, the solution builds the fabrication task, maintaining the referential integrity of data based on database constraints or applied constraints. Here, the constraints are solved by the solver and the solution is used to obtain the fabricated dataset [1]. The output can have multiple formats/extensions.

In order to generate fabricated data, we need to provide the constraints of the variables within the domain including the data fields and ranges of values. After specifying the constraints, the solver finds solutions by constraint propagation and search. Every time the solver generates a solution to all given rules (constraints), this solution is an instance in our dataset. Running the solver an indefinite number of times will give us a fabricated dataset which satisfies all the provided constraints. In case of inconsistencies in rules, no solution can be generated, but it indicates which rules are in conflict and these can be corrected.

Rules are formulated following the syntax accepted by the solver, which includes conditions, dependencies between fields, mathematical equations, ordering, and Boolean conditions. We can express weighted/probability, normal, and random distributions to determine the value of our fields.

Rules may result from a combination of medical knowledge and information extracted from the real dataset. Consider the rule below. If it is known that the cancer has metastasised into site *C34.9* we set *pulmonary_flag* to 1. Otherwise, we use a weight distribution to set the value of *pulmonary_flag*. We infer the weight distribution from the data extraction.

```
general.pulmonary_flag = (
  // Knowledge:
  (general.metastasis1 == 'C34.9' || general.metastasis2 == 'C34.9'
          || general.metastasis3 == 'C34.9') ? 1 :
  // From extraction:
  randomWeightedValue(general.pulmonary_flag,1200? 0, 120 ? 1). )
```

In Scotland, patients have a unique identifier given by the Community Health Index (CHI). The CHI has 10 digits consisting of the date-of-birth (*DDMMYY*) followed by a three-digit sequence number and a check digit. The ninth digit is always even for females and odd for males. To generate a proper CHI for patients we have to model this definition through several rules. For instance,

```
  allDiff(from(general), general.chi)
```

specifies that every CHI is unique. The next rule specifies the structure of a CHI,

```
  general.chi = concat(dateToString(general.DOB,DMy),
    intToString(general.D7),intToString(general.D8),
    intToString(general.D9),intToString(general.D10))
```

where the last four digits follow specific constraints. Here `D7,D8,D10` are arbitrary, e.g., `0 <= general.D7 <= 9`, and `D9` is used to indicate gender, which in our case has a 0.99 probability of being female given by:

```
 randomBool(99)?general.D9 = {0,2,4,6,8}:general.D9 = {1,3,5,7,9}
```

We specify the first *incident date* or diagnosis date, to be between 2014 and 2016 by using the *equality-inequality* relation:

```
  currentDate-(6*365)<general.incidence_date<= currentDate-(4*365)
```

We can use regular expressions to capture a postcode, and assign constants to fields such as cancer site, `general.site = 'C50.9'` to indicate breast cancer. We perform summation to populate the Charlson Comorbidity Index [9]. There are field values which influence other field values, and can be captured through implication (if there is not a first metastasis there cannot be a second or third).

```
(general.metastasis1 is Null -->> general.metastasis2 is Null)
  ->> general.metastasis3 is Null
```

To populate some fields we check whether we can use a normal distribution or add another correlation between fields from inspecting the original dataset. For instance, for the BMI we use the probability distribution to determine the category (e.g. underweight, normal, overweight) and then use a normal distribution to populate the exact BMI value for the patients in each category.

Some patients may have more than one hospital admission (recorded in the dataset *SMR01*) during their cancer care, for example, when they experience side-effects as a result of their treatment. Here, we create a new table for the patient admission and use the CHI as a reference to the general table. First, we specify the admission rate to fabricate the admission data. The rule is as follow:

```
numOf(from(smr01s), smr01s.chi = general.chi) =
  randomWeightedNumber(500 ? 1,300 ? 2,200 ? 0)
```

stating that 50% of patients have one admission, 30% have two and 20% patients have none. Since the admission date is time based (sequential), we create another helper field, *elapsed_days*. The admission date depends on both.

```
smr01s.admission_date=(smr01s.incidence_date + smr01s.elapsed_days)
```

The *elapsed_date* has a monotonically increasing value as follows:

```
monotonic(from(smr01s), per(smr01s.chi), smr01s.elapsed_days,
  {normalDistributionNumber(110.4, 17.2)}, randomNumber(14,100))
```

The first value for *elapsed_days* is populated using a normal distribution with 110.4 as the *mean* and 17.2 as the *variance*. The next instance of *elapsed_days* increases by a random number between 14 to 100 days. Because the admission date is calculated by adding *elapsed_days* to *incidence_date*, its value increases sequentially. With this, we can fabricate a patient's admission event.

The next dataset we fabricate is the chemotherapy treatment dataset, where the main challenge is capturing the relation between data that belongs to the same patient. Briefly, a patient may have more than one intention, and each intention may have more than one regimen. Each regimen has more than one cycle and so on. To capture this relation, we created five helper tables (i.e., *patients*, *intentions*, *regimens*, *cycles*, and *drugs*). There are similarities between these helper tables. We create *patients* as the reference point. We create *intentions* to model the condition where each patient may have one or more intentions. Similarly, *regimens* is created to model the condition where each intention may have one or more regimens (i.e., *cycles* and *drugs* have the same purpose). Each helper table has foreign keys to each other (e.g., *patient_id*, *intention_id*).

The *patients* table has the demographic information during the treatment, with values assumed to be relatively constant, such as *CHI*, *height*, *hospital*, *tumour_group*. The *patients* table acts as the proxy to the *general* where *CHI* is used as the foreign key. The ratio between the data in the *patients* and *general* table is set to one. We also have the *first_intention* field in this table, used as the reference for populating the intention value. We use *randomWeightedValue* to populate this field. By counting the number of each intention occurring in the first cycle, we can get the weight value. The rule for the *first_intention* is shown below:

```
    patients.first_intention = randomWeightedNumber(
      350? s'Adjuvant',
      200? s'Palliative',
      180? s'Neo-Adjuvant',
      15? s'Durable Remission',
      5? s'Curative')
```

The ratio between the *patients* and the *intentions* tables is determined by the *first_intention* because some intentions may or may not have follow up treatments. We specify the *intentions.ratio* rule as follow:

```
numOf(from(intentions),
  intentions.patient_id = patients.patient_id)= (
    intentions.first_intention == 'Adjuvant' ?
      randomWeightedNumber(15? 2: 1),
    intentions.first_intention == 'Durable Remission' ? 1,
    intentions.first_intention == 'Neo-adjuvant' ?
      randomWeightedNumber(60? 2: 1),
    intentions.first_intention == 'Palliative' ? 1,
    intentions.first_intention == 'Curative' ? 1)
```

The *first_intention* field determines the value of the next instance of *intention*. Similarly to the *patients*, we have a field *first_regimen*. The value of this field depends on *intentions.intention* and has the same function like the field *first_intention* (i.e., this method is repeated to capture the sequence behaviour for *cycles* and *drugs*).

To populate the treatment appointment date we use a similar rule (as for instance for patient hospital admission) as mentioned before. We have the *appointment_date* field in *intentions* to populate the first *appointment_date* for each intention. In the *intentions* table, we set elapsed days based on the regimen ratio, cycle ratio and regimen interval days to prevent the overlap between appointment dates for each regimen.

In the regimens, we have another *elapsed_day* field to determine the date of the first regimen. The starting date for the *regime.elapsed_day* is taken from the *regimen.init_appointment_date*. The *regimen.init_appointment_date* equals the *intention.appointment_date*. The rule for the *regimen.elapsed_day* is as shown below:

```
monotonic (from (regimens), per(regimen.intention_id),
  regimen.elapsed_days, (cycle_ratio * regimen_interval_days),
    {regimen.init_appointment_date})
```

Unlike *intentions* and *regimens*, we have the *cycle_ratio* in the *regimens* because we need to know the number of cycles for determining the correct elapsed days between regimens.

Finally, to populate several fields like the toxicity outcome, *regimens* and *performance status*, we integrate a simple Markov model into the rules (the value of the current fields depends only on its previous value). We use the previous value because we have observed a high correlation, based on the Pearson standard of correlation [3], between the previous value and the current value.

4 Conclusions and Future Work

We presented some of the rules describing the characteristics of our cancer treatment dataset which are fed to the IBM Data Fabrication Platform to generate synthetic data. The rules describe the expected range of values within a column, relationships between columns, and - more significantly - relationships between rows where these describe different events in the treatment of the same patient. An accurate set of rules is essential to generate realistic data, and we need to evaluate how realistic the synthetic data is. Machine learning can be useful to establish this to some extent, but was outside the scope of the present paper.

Although synthetic data is valuable it is not a replacement of real data. If all the features present in a dataset have been incorporated into a synthetic dataset, then the later may in fact have the same biases as the original dataset. However, we believe that an added advantage of using the IBM Data Fabrication Platform comes from the ability to generate rules derived from a combination of domain knowledge directly (in our context this includes information from clinical guidelines, clinical studies as well as medical practice) and features extracted from real data. This flexibility, may consequently lead to a synthetic dataset less prone to biases inherent in real data specially when real datasets are small.

References

1. Adir, A., Levy, R., Salman, T.: Dynamic test data generation for data intensive applications. In: Hardware and Software: Verification and Testing (HVC 2011). LNCS, vol. 7261, pp. 219–233. Springer (2011)
2. Adorf, H.M., Varendorff, M.: Constraint-based automated generation of test data. In: Software Quality: Model-Based Approaches for Advanced Software and Systems Engineering (SWQD 2014). LNBIP, vol. 166, pp. 199–213. Springer (2014)
3. Akoglu, H.: User's guide to correlation coefficients. Turk J Emerg Med **18**, 91–93 (2018)
4. Caiola, G., Reiter, J.P.: Random forests for generating partially synthetic categorical data. Transactions on Data Privacy **3**, 27–42 (2010)
5. Dandekar, A., Zen, R.A.M., Bressan, S.: Comparative evaluation of synthetic data generation methods. In: Proceedings of ACM conference (Deep Learning Security Workshop) (2017)
6. Janic, V., Bowles, J.K.F., Vermeulen, A.F., et al.: The Serums tool-chain: Ensuring security and privacy of medical data in smart patient-centric healthcare systems. In: IEEE International Conference on Big Data (IEEE Big Data 2019) (2019)
7. de Moura, L., Bjørner, N.: Satisfiability Modulo Theories: Introduction and applications. Communications of the ACM **54**(9), 69–77 (sep 2011)
8. Reiter, J.P.: Using CART to generate partially synthetic public use microdata. Journal of Official Statistics **21**, 441–462 (2005)
9. Roffman, C.E., Buchanan, J., Allison, G.T.: Charlson Comorbidities Index. Journal of physiotherapy **62**, 171 (2016)
10. Rubin, D.B.: Discussion statistical disclosure limitation. Journal of Official Statistics **9**, 461–468 (1993)
11. Silvina, A., Bowles, J., Hall, P.: On predicting the outcomes of chemotherapy treatments in breast cancer. In: Artificial Intelligence in Medicine. LNCS, vol. 11526, pp. 180–190. Springer (2019)
12. Silvina, A., Bowles, J., Hall, P.: Combining patient pathway visualisation with prediction outcomes for chemotherapy treatments. In: 12th International Conference on eHealth, Telemedicine, and Social Medicine. pp. 108–113. IARIA (2020)
13. Tsang, E.: Foundations of Constraint Satisfaction. Academic Press (1993)