Achieving Privacy-Utility Trade-off in existing Software Systems

Saurabh Srivastava, Vinay P. Namboodiri, T.V. Prabhakar

Department of Computer Science & Engineering, IIT Kanpur, India - 208016

ssri@iitk.ac.in, vinaypn@iitk.ac.in, tvp@iitk.ac.in

Abstract. Privacy and utility of data are two aspects of a system that are often diagonally opposite to each other. Privacy concerns drive design decisions that can reduce the ability to make deductions or correlations from a given dataset (e.g. reducing the probability that an individual could be recognised from a given set of health records). Utility, on the other hand, tries to maximise the chances of finding essential relationships in the real world, that can then be used for making smarter systems (e.g. the ability to predict that an individual is at higher risk of being affected by a terminal disease). A term that is often used to explain this paradox is called the Privacy-Utility trade-off. Software practitioners have often ignored the privacy aspects due to lack of legal obligations, and have generally concentrated on achieving functionality. But with a renewed interest in Artificial Intelligence, privacy concerns are going to become more critical in the near future. This will force the software providers to reevaluate their existing products and services from a privacy perspective. In this work, we analyse some of the challenges that a typical software provider would face while doing so. We present a privacy model that can be applied to existing systems, which in turn can suggest first-cut privacy solutions requiring minimal alterations in deployed applications. To the best of our knowledge, no open-source initiative has been started until now to cater to these requirements. We briefly introduce the prototype of an open-source tool that we are developing, which is aimed at facilitating this analysis. The initial results were obtained over some standard datasets, as well as a real-world credit card fraud dataset, which seemed to collate with our intuitions.

1. Introduction

Protecting the privacy of an individual in a dataset is a problem that has been studied over the years by many researchers. This problem becomes even more challenging in the context of AI applications, which generally require a substantial amount of data to be able to achieve real- world objectives. For instance, as the May 2018 implementation date of the EU General Data Protection Regulation (GDPR)[1] approached, panic struck the software providers in Europe and they were forced to dedicate more time and effort towards honouring privacy concerns in their products and services[2].

While there is no universally accepted privacy definition, the one that has been followed the most is the notion of *Differential Privacy* [3]. In a nutshell, differential privacy attempts to provide probabilistic guarantees to a user, that provided there is a *substantial crowd*, determining the presence or identifying the records of an *individual* is unlikely. While the concept may be sound and backed up by elegant theoretical analysis, understanding the nuances of its application to a real-world problem is not trivial.

Another approach aimed towards involving practitioners in the industry is the general concept of *Privacy by Design* (PbD)[4]. The approach attempts to provide a collection of principles to keep privacy concerns in the loop while designing a system. Unfortunately, there is still a gap between the principles themselves and the detailed guidelines on how to achieve them in practice. A major reason behind this is a lack of detailed mappings between the principles and the tools or practices that achieve them. Our work provides practitioners with ways to achieve the *Privacy as the Default Setting* principle[4], specifically, relating to the FIPs *Collection limitation* and *Data minimization*[4].

In this work, we provide a simple model for privacy, assuming that a software provider may not be well versed in the intricacies of the modern day privacy approaches. It must be noted that our model works as a *supplement* to the existing privacy techniques, and in most cases, can be used as a first-cut towards achieving privacy in practice, while the sophisticated privacy- preserving algorithms can be applied later on. The motivation of our model comes from *wrapper feature selection*[5], a concept often used by Data Scientists to reduce the dimensionality of a dataset while considering the inherent system as a black-box. As long as the learning algorithm being used can provide a performance value for different learning tasks, wrapper methods can be applied. Although there are inherent issues with these methods from the perspective of computational cost, these methods can be used with almost any learning activity, since they do not use any algorithm specific properties. For evolving a privacy model that needs to be applied across varying software systems and services, these methods can yield practical solutions that require minimal changes in the existing systems. Almost all the existing privacy-preserving techniques can then be applied, if needed, to achieve even higher levels of privacy.

To summarise, we attempt to answer the following questions:

(i) Can we build a generic privacy model that can be applied to a wide range of data usage mechanisms?

(ii) Is it possible to build a privacy framework that is more suited for application to existing systems, with very few assumptions about how they work or what they do with data?

(iii) Does comprehension of such a privacy model requires the skills of a Data Scientist? Considering that many software practitioners like a Subject Matter Expert (SME), Functional Analyst (FA), Solution Architect (SA) or a Developer (Dev) [6] may not have expertise in Information Science, can they use the model in practice without requiring an introductory course in Probability and Statistics?

(iv) Are there any existing (preferably free) privacy analysis tools that can be used by a practitioner to analyse privacy concerns for her system? If so, can these tools provide basic suggestions on how to achieve the privacy-utility trade-off that we discussed before?

The rest of this paper is structured as follows. In Section 2, we cover a short survey of selected work in the field of privacy techniques. This builds up the motivation for the current work. In Section 3, we provide a detailed explanation of a privacy model that attempts to cater to the questions raised above. In Section 4, we provide details of how a working prototype of the model can be accessed and extended. We also provide an abstract of our observations after using the tool over some datasets. Finally, we culminate the discussion in Section 5 by concluding our work and iterating over the future aims to achieve.

2. Related work

We discuss some of the work, that is relevant to our current efforts. The work of our interest can be divided into three major (not necessarily exclusive) categories.

First, a significant amount of work has gone towards building differentially private formulations of some machine learning techniques. Attempts to do so for SVM[7], linear and logistic regression[8], bayesian detection[9], nearest neighbor classification[10], random forests[11] etc. have been made in the past. Similar attempts have also been made for even deep learning methods[12]. Generic methods like “input perturbation” or “output perturbation” have also been tried[13]. These works are of use to a practitioner, in case these works are turned into (preferably free and open-source) off-the-shelf software components. Such components should also be easy to integrate with existing applications or

for use as a "black-box" in a new project. Unfortunately, most of them were not taken up to that level, and are available mostly as demo applications only. There are some important initiatives towards bridging this gap [14][15][16][17]. However, more such efforts are necessary to make privacy solutions easy to integrate within a software project.

Another set of previous works that are of interest to us involve the concept of *anonymisation*. It is an attempt towards incorporating differential privacy in a realistic setting. The most common example of such a technique consists of the use of "generalisation" and "suppression" of data values to achieve k -anonymity[18]. There are several versions (datafly[19], mondrian[20], incognito[21] etc.) and refinements (l -diversity [22] and t -closeness[23]) of k -anonymity over a period of time. These methods can be applied over a specific use-case with the use of the **UTD Anonymization Toolbox**[24]. However, a practical limitation of the tool when implementing k -anonymity involves a complicated phase of defining a "taxonomy tree" for nominal attributes. This is a precursor to the algorithm climbing up in the tree to generalise data items (e.g. values like "Local Government", "Federal Government" or "State Government" for an attribute *workclass*, could be generalised to a more general value like "Government"). Two different ways to generalise the values can be expressed by providing different trees. This, in turn, can yield different anonymization of the same dataset, honouring the equal values of k. For a practitioner, this uncertainty would require additional effort to reach a consensus regarding the taxonomy to use for anonymisation, making it impractical for production use.

Lastly, there are relatively recent works, which may be considered as actual precursors to our work, which explore the idea of *privacy-aware* feature selection ([25][26][27][28][29][30] etc.). We can contrast our work from the discussed sources, in one or both of the following manner. First, the model presented in our work doesn't expect any background knowledge in the field of Statistics, Probability or Information Theory, for comprehension. The model uses metrics that a software practitioner might already be aware of (e.g. "Classification Accuracy"), rather than defining any new privacy metric needing additional reading. It should be put in perspective, since understanding a custom privacy metric might be too cumbersome for software practitioners. Second, we have started a project aimed at building a holistic engineering solution which can be used to examine a variety of datasets, with very little pre-processing. Our efforts are based on the general *convention over configuration*[31] philosophy, including ways to select default values for most parameters, meaning that the solution doesn't require a steep learning curve. To the best of our knowledge, our work is the first to address these concerns.

3. The trade-off model

To comply with our assumptions that a practitioner will use the model, we provide a semi- formal approach for analysing privacy, which can be easily assimilated without any background knowledge. Theoretical insight into how feature selection can be used as a measure for privacy

age	marital-status	race	*class*
35	Divorced	White	*>50K*
38	Divorced	White	*<=50K*
53	Never-married	White	*<=50K*
49	Married-civ-spouse	Black	*<=50K*
42	Married-civ-spouse	White	*>50K*

age	workclass	*class*
53	Local-gov	*<=50K*
28	Private	*<=50K*
35	Private	*>50K*
37	Private	*<=50K*
39	State-gov	*<=50K*
49	Private	*<=50K*

race	*class*
White	*<=50K*
Black	*<=50K*
White	*>50K*
Other	*<=50K*

age	*class*
37	*>50K*
49	*>50K*
38	*<=50K*
42	*>50K*
38	*>50K*

Figure 1. Some partitions of the dataset in Table 1

has already been covered in previous works like [25][27][32], which can be referred for gaining more insights into our approach. We focus our attention on using the idea to build a model that is simple and intuitive and does not require any prerequisites for a software practitioner in the field of Data Science or Information Theory.

3.1. Problem Formulation

Simply put, the reason that privacy and utility are at loggerheads with each other for any data-intensive activity is that they intend to achieve goals that are diagonally opposite to each other. In abstract terms, *Utility* in the context of any learning activity is about *finding correlations* in a real-world scenario from the given data. The examples include grouping data items into similar categories (unsupervised clustering), assigning them to one of the given classes (classification) or guessing an unknown value based on some observations (regression). The utility of a dataset in this respect is the ability to be able to find better correlations. On the other hand, in the context of any data collection activity *Privacy* is about *hiding correlations*. The examples could be removing personally identifiable information from the dataset, and more importantly, dissolving the correlations among the quasi-identifiers[33], to whatever extent it is feasible. This means that for any practical usage, a trade-off needs to be achieved between the two based on real-world scenario [34][35]. To better understand this trade-off, consider the case of a small dataset with 12 rows and 5 columns, as shown in Table 1.

This sample has been taken from the UCI Adult dataset[36]. The example dataset has four quasi-identifiers (we assume that any personal identifiers are already removed) **age**, **workclass**, **marital-status** and **race**. The fifth attribute is the ***class*** attribute, which assigns a "label" to the rows, putting them in one of the two income groups *<=50K* or *>50K*.

Now consider a scenario where we break the above dataset, into two or more different **partitions**, each containing one or more of the quasi-identifiers as well as the class attribute1, having all or some of the original rows, not necessarily in the same order. Examples of partitions for the dataset in Table 1 are shown in Figure 12.

[1] From here on, whenever we use the term attribute, it means any attribute other than the **class** attribute.

[2] From here on, we omit the **class** attribute while showing a partition since its presence is implicit.

Table 1. An excerpt from the UCI Adult dataset

age	workclass	marital-status	race	*class*
39	State-gov	Never-married	White	*<=50K*
49	Self-emp-inc	Married-civ-spouse	White	*>50K*
28	Private	Married-civ-spouse	Other	*<=50K*
35	Private	Divorced	White	*>50K*
38	Private	Divorced	White	*<=50K*
53	Local-gov	Never-married	White	*<=50K*
28	Private	Married-civ-spouse	Black	*<=50K*
37	Private	Married-civ-spouse	Black	*>50K*
37	Private	Married-civ-spouse	White	*<=50K*
49	Private	Married-spouse-absent	Black	*<=50K*
38	Federal-gov	Married-civ-spouse	White	*>50K*
42	Private	Married-civ-spouse	White	*>50K*

We can make two important observations by seeing the original dataset, and some of its partitions. If any real-world correlations existed between two or more attributes, that can help an adversary discover a piece of information that is critical from a privacy perspective, then putting them into different partitions, will imply that the *privacy* of a partitioned dataset is *higher* than that of the original dataset. Second, by selecting only a fraction of the overall rows in the original dataset and/or

jumbling their order, it becomes quite tricky (may even be impossible) to regenerate a complete row, given some of the partitions. This implies that the *utility* that a partition provides may be *lower* (or at least not higher) than the original dataset. This assumption is rather weak, as it is possible that a partition may have higher utility than the complete dataset, because some irrelevant features may have been removed. Nevertheless, if we run into such a case, the model works even better, but for our discussions, we use a stricter assumption and assume that there are no irrelevant attributes in the dataset, and removing any attribute cannot increase the utility of the learning activity.

To reiterate, the highest privacy can be achieved, if we create partitions that have just *one attribute* out of all (along with the class attribute), and the highest utility may be made if we create a partition with *all the attributes*. With these observations, we can now construct a privacy-utility trade-off model. The general idea is that by choosing an "appropriate" granularity for breaking up the original dataset, we can achieve varying levels of privacy. Such a model can be particularly helpful in cases where a learning task has the additional challenge of providing a near real-time output, e.g. Credit Card Fraud detection systems. The model can hence serve another purpose, that of a prospective thinning of the dataset for faster predictions.

It must be noted here that this doesn't necessarily mean that partitions of the "same size" provide the "same level of privacy", because certain attribute combinations may reveal more in reality as compared to others (we return to this discussion in Section 3.2). The only claim here is that given a partition, any partition which is its superset, cannot provide more privacy than the partition itself. This is based on the assumption that any correlations that can be found by an adversary in the original partition can still be found by ignoring the extra attributes. Alternatively, we can claim that any partition that is a subset of a given partition, cannot provide lesser privacy than the partition itself, by a similar argument.

If we go by the above explanation, to achieve high levels of privacy, we will have to be content with low levels of utility. Thankfully, it may not be the case always. Feature Selection, a process aimed at "selecting the most relevant features" (or "trimming the least relevant features") of a dataset, is a technique often used by data scientists before applying any machine learning tasks over it. The rationale is different though from our problem. It is usually done to achieve faster learning time or accomplish the task over limited infrastructure by providing only those features to the algorithm, which could be more effective. In some instances, it may even improve the accuracy of the algorithm, because some of the features that were removed, may have just been "noise". This motivates to attempt feature selection over a dataset, with the primary objective being preserving privacy, rather than achieving the goals mentioned above. We can now try to formalise our observations in the form of a simple privacy model.

3.2. Trade-off Parameters

The most important parameter in this trade-off model is the **partition size**, i.e. the number of attributes in the partition. Clearly, this number lies between 1 and ***n***, where *n* is the *total number of attributes in the dataset, minus the class attribute*. This parameter being closer to 1 means a slide towards the privacy side of the trade-off, and proximity to *n* shows a tilt towards utility.

One practical aspect that our simplistic formulation doesn't cater to is the fact that in a real world scenario, not all attributes may have equal significance in terms of privacy or utility. For instance, in our sample dataset, it could be possible that a partition containing both age and workclass could pose a higher risk to privacy in the real world than say another partition that has age with marital-status. The specific sets of attributes that are more "sensitive" for being grouped depend on the particular use-case, as well as the adversarial behaviour, against which the software provider wants to protect the application.

This leads to the addition of another composite parameter in our model called **privacy exceptions**. Privacy exceptions are a (possibly empty) list of attribute sets, that should never be put together in any partition that is to be considered for use in production. In the software development process, these

exceptions should ideally be provided by Subject Matter Experts (SMEs), Functional Analysts (FAs) or Solution Architects (SAs) [6].

Another core parameter to our model is the **learning objective**. This parameter defines the actual purpose for which this dataset is to be used. At first, this may seem contrary to one of our fundamental goals - to have a model that is independent of the actual learning activity that a software component performs. However, the model does not depend on the learning objective itself. It is a parameter that is used for running *experiments* (we have covered in detail, what an experiment looks like in [37]), and in practice, any machine learning task can be used as a learning objective. The only requirement that the model has is that the learning mechanism can produce an **output metric**, that can be compared across multiple partitions of the dataset. For example, the most common metric for classification problems is the "Classification Accuracy". In theory, any metric that the learning component provides can be used in our model. There are some additional parameters too that we added to our model during the implementation of a prototype to make the model practically viable. We now describe how a solution to the overall problem can be found using an engineering approach.

3.3. Selecting a Partition

Even if the values for the above parameters are fixed, the number of partitions that will satisfy the criteria will almost always be more than one. For the sample dataset, if we choose the following parameters:

partition size = 2;
privacy exceptions = { (age, workclass) };
learning objective = Classification(NaiveBayes);

it is interpreted as:

*"Choose a partition having **2** attributes for **Naive Bayes Classification**; while choosing the solution, don't consider partitions that have **age** and **workclass** together."*

It can be noticed that we have not explicitly mentioned the metric that would be used. We adopt the *convention over configuration* [31] philosophy for our model. This means we pick *default* parameters, wherever possible, and allow them to be overridden, if necessary. We consider the classification accuracy as the default metric produced by any classification component.

The partitions that follow these criteria are:

(i) {age, marital-status}
(ii) {age, race}
(iii) {workclass, marital-status}
(iv) {workclass, race}
(v) {marital-status, race}

Clearly, the simple model that we've presented till now will have no way to differentiate between the above partitions, since they all provide the same level of privacy. At this point, we stop with the conceptual modelling and enter the realms of engineering. We order these partitions by a metric that the learning module provides on executing it over the partitions. For our sample case, if we create these 5 partitions of the dataset, with all the rows, and run the Naive Bayes Classifier over them, we get the following values for the Classification Accuracy metric:

{age, race}	**58.333333333333336%**
{age, marital-status}	33.333333333333336%
{workclass, marital-status}	33.333333333333336%
{marital-status, race}	33.333333333333336%
{workclass, race}	25.0%

It seems that using {age, race} could provide a better utility, with the same level of privacy, for this particular use-case. While in this case, the decision to pick a specific partition seems rather

straightforward, it is relatively common to end up with the same value of a metric for multiple partitions, as shown for three of out of the five partitions above. This means that there may be more than one partition, that can be chosen for practical usage, provided they give the highest (or reasonably high) value of the metric. The prototype tool that we've built[37], shows the user "all the options" (or at least, a "satisfactory number of options") available for the solution, in sorted order of a chosen metric, allowing the user to take the final call for using in production. Essentially, the tool shows the user how various partitions of the original dataset would probably fair in a real-world scenario, and aids in the process of finalising one or more partitions to use in production.

3.4. Additional Parameters

There is a significant practical limitation with this model if we run into a dataset that is "too fat" (has too many attributes) or "too tall" (has too many rows). This is because iterating over all possible partitions of a dataset, and running the learning module over them, will become computationally too intensive to afford. The function that actually determines the number of partitions is the Combinations function, $C(n, k)$, which is not linear. For instance, with $n = 25$, the value of $C(n, k)$ jumps from 53,130 for $k = 5$, to more than 3 million (3,268,760) for $k = 10$. This means that for even a moderately large value of n, the number of partitions to try out with the learning module might soon exceed limits of practicality.

To counter that, we add two additional parameters to supplement the model. The parameters **vertical expense** and **horizontal expense** can respectively limit the *number of combinations that are tried* and *the number of rows that are placed in any partition*. This is an engineering trade-off, that depends on the actual hardware (or virtual hardware) resources that are used to

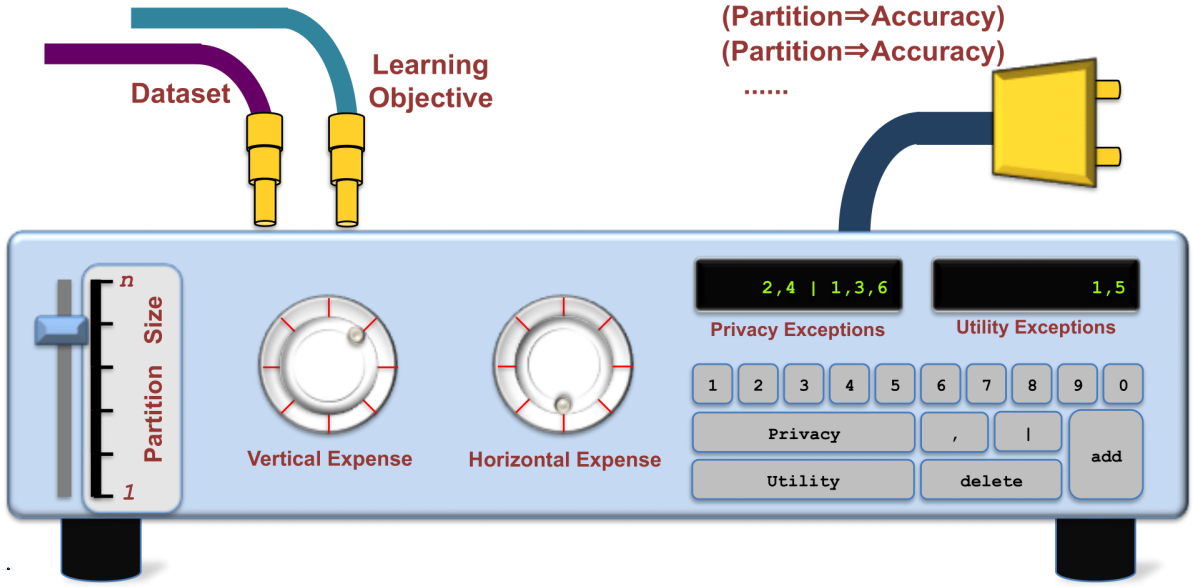


Figure 2. An Overview of the privacy model: (i). Dataset and Learning Objectives are fed as inputs to the model. (ii). Privacy and Utility Exceptions are defined in terms of Column Numbers in the original dataset. (iii). Partition size, Vertical Expense and Horizontal Expense are configurable parameters. (iv). The output of the model is a set of partitions along with the accuracies they achieved over sample data.

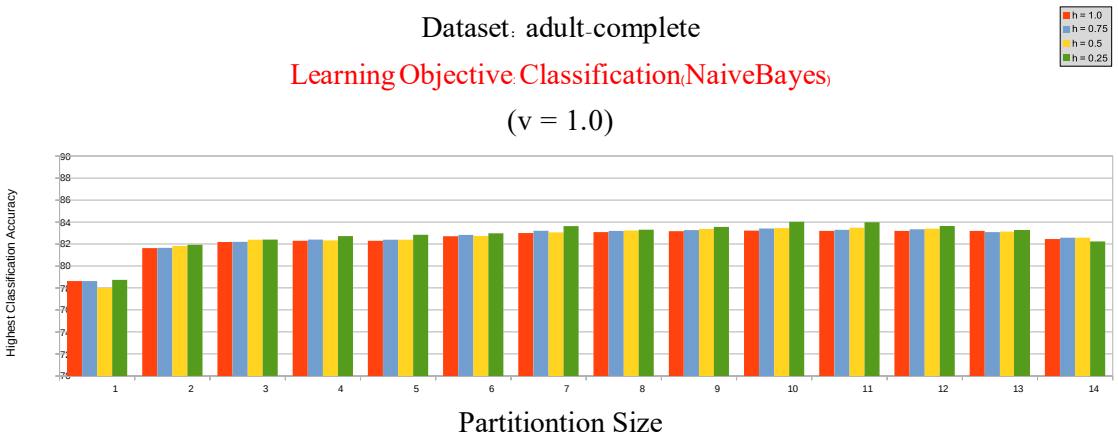
run the experiments. These engineering trade-offs are often required in certain real-world cases, where soft real-time guarantees are expected from a prediction component (such as classifying a credit card transaction as fraudulent and denying it within a few seconds after the card swipe). A value of "0.5" for the vertical expense, means that only half of the possible partitions should be tried out.

Similarly, choosing a value of "0.5" for horizontal expense indicates that the number of rows to be put inside a partition should be half of those in the original dataset.

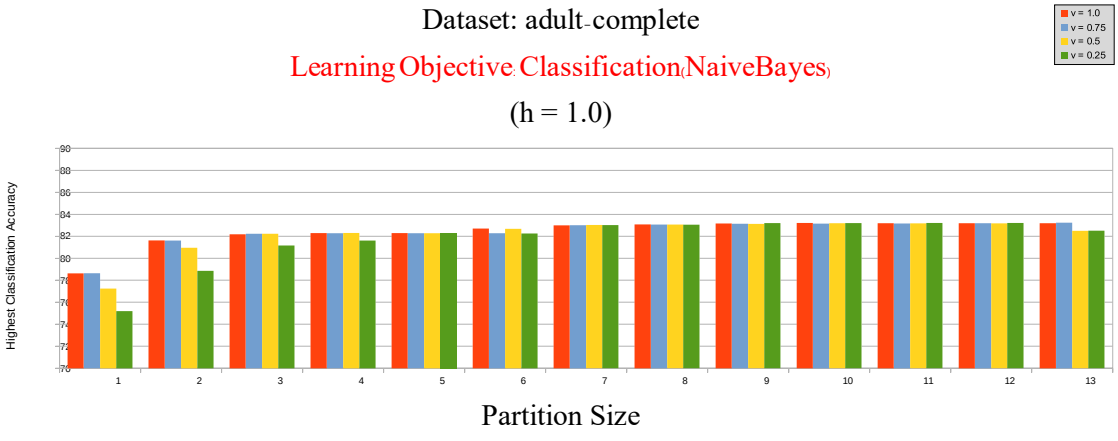
This brings us to the last additional parameter that we add to our model. If the vertical expense is set to any value less than 1, it would mean that at least some prospective partitions, will have to be left out at the experiment stage. The last knob of fine-tuning that a practitioner would like to be at her behest would be to prevent certain partitions from the axe. To do so, we provide a parameter which is similar to privacy exceptions, called **utility exceptions**. Utility exceptions, just like privacy exceptions are a (possibly empty) list of attribute combinations, that the practitioner always wants to be considered for the execution stage of the experiment (to the extent it is possible). This means, if some partitions are to be discarded, those that contain some utility exception, are more immune than those who don't. We must clarify here that in case there is a conflict between a privacy and a utility exception, i.e. there is a partition that contains at least one of each, then the **privacy exception takes precedence over the utility exception**, and the partition is not considered further. In essence, while ***privacy exceptions*** **are "constraints"** on the model, the ***utility exceptions*** **are only "advises"** . Figure 2 shows a visualisation of the model, showing the inputs, the output, as well as the parameters.

4. Observations from prototype implementation

We have built an open-source tool as a prototype to initiate efforts in the dimension of building practical tools for software practitioners, who would wish to analyse the usage of different types of datasets, currently in use. We have shown the working of the tool; it's GUI as well as command-



(a) Varying horizontal expense, keeping vertical expense constant



(b) Varying vertical expense, keeping horizontal expense constant

Figure 3. Results for the UCI Adult dataset with learning objective: Classification (NaiveBayes)

line interfaces, as well as results of experiments done on different datasets[37]. [37] shows the operation of the tool; it's GUI; it's Command-line interface as well as some preliminary results.

We had analysed the results of the tool over different formulations of the UCI Adult dataset[36] as well as a real-life credit card fraud dataset[38]. The results over both the datasets were encouraging[37]. The details of the experimental setup, as well as the numerous experiments that we performed, are discussed in[37], here we attempt to provide logical observations from the experiments helpful in understanding the relevance of the model presented in Section 3. Some results from the experiments over the complete UCI Adult dataset are reproduced in Figure 3 for a better understanding of our observations. The results can be abstracted to the following comments:

(i) In almost all cases, the tool was able to find at least one partition, which involved a fewer number of attributes than the original dataset, but the change in accuracy (or some other metric) was not drastic.
(ii) In some cases, the partitions performed even better than the original dataset3. This is a common observation in Feature Selection process, but the difference here is that the columns chosen (or not chosen) for the purpose were driven by a privacy-first approach, rather than prioritising accuracy.
(iii) The set of all partitions possible for a dataset may be too large to try out with a brute-force approach. The tool was able to randomly select a small percentage of overall partitions

[3] Implying maximum values for a partition size, vertical as well as horizontal expenses

(by variation in vertical expense), while using only a fraction of the complete dataset (by variation in horizontal expense) and was still able to suggest good alternatives to using the complete dataset. This implies that while the best possible partition might not have been tried by the tool, a "practically acceptable" solution can still be found.

These observations are concurrent with almost all the experiments that we performed with the tool. This was encouraging because the results were in line with our intuitions while discussing the model.

The tool is already available under the MIT license[39] for evaluation. There is a demo video available too for any beginners wanting to try out the tool and contribute towards the project[40]. The tool is duly supported by a User Manual, which contains details for all the switches as well as two auxiliary utilities, bundled with the main tool.

5. Conclusion and future scope

In this work, we presented a model for analysing the trade-off between privacy and utility in existing systems. We claimed that by varying the size of a dataset, we could achieve varying levels of privacy and utility. We argued that our model can be helpful for a typical software practitioner since it neither requires any backgrounds in Data Science or Information Theory nor does it seek to modify existing learning components before use. We must reiterate that this model helps provide a first-cut solution only, as it attempts to reduce the overall collected data, without hampering the utility of the application or service. If the privacy requirements for applications are more stringent, other sophisticated privacy techniques can then be applied over a dataset suggested by this model.

We also briefed about a prototype open-source tool called **PUTWorkbench**[37] used to verify the claims made above, and the overall results were encouraging. The tool that comes with a command-line interface for scripting and automation, as well as a GUI to analyse the effects of individual experiments via an interactive display is available for evaluation and contribution from the open-source community.

In future, we would like to develop the tool into a full-fledged, industry-ready privacy analyser. This tool would contain possibilities to apply one or more privacy techniques over a dataset in sequence, and provide the user with an easy to use interface to analyse the results.

Another exciting possibility to explore is if a similar model can be evolved for use-cases involving Deep Learning [41], where the number of attributes can range in thousands, and applying this model becomes practically intractable.

References

[1] Regulation G D P 2016 *Official Journal of the European Union (OJ)* **59** 1–88
[2] 2018 GDPR: The great data privacy panic URL https://www.bbc.com/news/technology-44240664
[3] Dwork C 2006 Differential privacy *Proceedings of the 33rd International Conference on Automata, Languages and* Programming - *Volume Part II* ICALP'06 (Berlin, Heidelberg: Springer-Verlag) pp 1–12 ISBN 3-540- 35907-9, 978-3-540-35907-4 URL http://dx.doi.org/10.1007/11787006_1
[4] Cavoukian A 2009 *Take the* challenge. *Information and privacy commissioner of Ontario, Canada*
[5] Kohavi R and Sommerfield D 1995 Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. *KDD* pp 192–197
[6] Bogue R 2005 Cracking the code: Breaking down the software devel- opment roles URL https://www.developer.com/mgmt/article.php/3490871/Cracking-the-Code-Breaking-Down-the-Software-Development-Roles.htm
[7] Rubinstein B I, Bartlett P L, Huang L and Taft N 2009 *arXiv preprint arXiv:0911.5708*
[8] Zhang J, Zhang Z, Xiao X, Yang Y and Winslett M 2012 *Proceedings of the VLDB Endowment* **5** 1364–1375
[9] Li Z and Oechtering T J 2015 *IEEE Journal of Selected Topics in Signal Processing* **9** 1345–1357
[10] Gursoy M E, Inan A, Nergiz M E and Saygin Y 2017 *Data Mining and Knowledge Discovery* **31** 1544–1575
[11] Fletcher S and Islam M Z 2017 *Expert Systems with Applications* **78** 16–31
[12] Abadi M, Chu A, Goodfellow I, McMahan H B, Mironov I, Talwar K and Zhang L 2016 Deep learning with differential privacy *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (ACM) pp 308–318*
[13] Sarwate A D and Chaudhuri K 2013 *IEEE signal processing magazine* **30** 86–94
[14] Mohan P, Thakurta A, Shi E, Song D and Culler D 2012 Gupt: privacy preserving data analysis made easy *Proceedings of the 2012 ACM SIGMOD International Conference on Management of* Data (ACM) pp 349–360
[15] McSherry F D 2009 Privacy integrated queries: an extensible platform for privacy-preserving data analysis *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data* (ACM) pp 19–30
[16] Roy I, Setty S T, Kilzer A, Shmatikov V and Witchel E 2010 Airavat: security and privacy for mapreduce *Proceedings of the 7th USENIX conference on Networked systems design and implementation* (USENIX Association) pp 20–20
[17] Katla S 2017 DPWeka*: Achieving Differential Privacy in WEKA* Ph.D. thesis University of Arkansas
[18] Samarati P and Sweeney L 1998 Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression Tech. rep. Technical report, SRI International
[19] Sweeney L 1998 Datafly: A system for providing anonymity in medical data *Database Security XI* (Springer) pp 356–381
[20] LeFevre K, DeWitt D J and Ramakrishnan R 2006 Mondrian multidimensional k-anonymity *Data* Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on (IEEE) pp 25–25
[21] LeFevre K, DeWitt D J and Ramakrishnan R 2005 Incognito: Efficient full-domain k-anonymity *Proceedings* of *the 2005 ACM SIGMOD international conference on Management of data* (ACM) pp 49–60
[22] Machanavajjhala A, Kifer D, Gehrke J and Venkitasubramaniam M 2007 *ACM Transactions on Knowledge Discovery from Data (TKDD)* **1** 3

[23] Li N, Li T and Venkatasubramanian S 2007 t-closeness: Privacy beyond k-anonymity and l-diversity *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on* (IEEE) pp 106–115
[24] 2012 UTD anonymization toolbox URL http://cs.utdallas.edu/dspl/cgi-bin/toolbox/
[25] Pattuk E, Kantarcioglu M, Ulusoy H and Malin B 2015 Privacy-aware dynamic feature selection *Data* Engineering *(ICDE), 2015 IEEE 31st International Conference on* (IEEE) pp 78–88
[26] Aristodimou A, Antoniades A and Pattichis C S 2016 *Healthcare Technology Letters* **3** 16–21
[27] Yang J and Li Y 2014 Differentially private feature selection *Neural Networks (IJCNN), 2014* International Joint *Conference on* (IEEE) pp 4182–4189
[28] Sheikhalishahi M and Martinelli F 2017 Privacy-utility feature selection as a privacy mechanism in collaborative data classification *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2017 IEEE 26th International Conference on* (IEEE) pp 244–249
[29] Lu Y, Yan M, Han M, Yang Q and Zhang Y 2018 *Mathematical Foundations of Computing* **1** 331–348
[30] Ravindran S and Aghila G 2019 A privacy-preserving feature extraction method for big data analytics based on data-independent reusable projection *Handbook of Research on Cloud* Computing *and Big Data Applications in IoT* (IGI Global) pp 151–169
[31] Bächle M and Kirchberg P 2007 *IEEE software* **24**
[32] Thakurta A G and Smith A 2013 Differentially private feature selection via stability arguments, and the robustness of the lasso *Proceedings of the 26th Annual Conference on Learning Theory* (*Proceedings of Machine Learning Research* vol 30) ed Shalev-Shwartz S and Steinwart I (Princeton, NJ, USA: PMLR) pp 819–850
[33] 2005 OECD Glossary of Statistical Terms - Quasi-identifier Definition URL http://stats.oecd.org/ glossary/detail.asp?ID=6961
[34] Li T and Li N 2009 On the tradeoff between privacy and utility in data publishing *Proceedings of the 15th* ACM *SIGKDD international conference on Knowledge discovery and data mining* (ACM) pp 517–526
[35] Erdogdu M A and Fawaz N 2015 Privacy-utility trade-off under continual observation *2015 IEEE* International *Symposium on Information Theory (ISIT)* (IEEE) pp 1801–1805
[36] Lichman M 2013 UCI machine learning repository URL http://archive.ics.uci.edu/ml
[37] Srivastava S, Namboodiri V P and Prabhakar T 2019 *arXiv preprint arXiv:1902.01580*
[38] 2018 Credit Card Fraud Detection URL https://www.kaggle.com/mlg-ulb/creditcardfraud
[39] Initiative O S *et al.* 2006 The mit license
[40] 2018 PUTWorkbench - A Quick look URL https://www.youtube.com/watch?v=xcPq8Y0ZeeM
[41] Schmidhuber J 2015 *Neural networks* **61** 85–117