Model Slicing for Responsible AI

Parke Godfrey
York University
godfrey@yorku.ca

Lukasz Golab
University of Waterloo
lgolab@uwaterloo.ca

Divesh Srivastava
AT&T Chief Data Office
divesh@research.att.com

Jarek Szlichta
York University
szlichta@yorku.ca

ABSTRACT

Model slicing refers to dividing a dataset into semantically meaningful subgroups (slices) and computing model performance metrics such as accuracy for each slice. Underperforming slices can then be used as starting points for investigating systematic issues with the model or the training data. We survey model slicing for responsible AI, with an emphasis on recent extensions to capture various types of training data and model bias.

VLDB Workshop Reference Format:
Parke Godfrey, Lukasz Golab, Divesh Srivastava, and Jarek Szlichta. Model Slicing for Responsible AI. VLDB 2025 Workshop: Guide-AI.

1 INTRODUCTION

Predictive models must be validated on testing data not seen during training. Alongside metrics such as the overall accuracy, it is important to inspect model performance on subsets of the testing set. These subsets may describe protected or minority subgroups in medical or financial models, or various weather or road conditions in autonomous driving models.

Assessing model performance across semantically meaningful subgroups is known as *model slicing* [3, 16]. Consider the testing set in Table 1, describing loan approvals. Each example consists of a surrogate key, *id*, followed by the model features (the applicant's sex, age, ethnicity and income), the ground truth label (one if the applicant repaid their loan, zero otherwise), and the model's prediction. Suppose that we group by sex, age and ethnicity to define the data cube of testing set slices, and suppose that we wish to view slices with at least four examples (i.e., support of at least four). Table 2 shows the results of model slicing, sorted in ascending order of model accuracy. Each slice is a conjunction of attribute-value pairs (or attribute-value-range pairs), with a star symbol representing a roll-up of the corresponding attribute. The overall model accuracy on this testing set is $\frac{13}{16} = 81\%$, reported in the all-stars slice, but the model underperforms for male applicants (the "M * *" slice, with 67% accuracy), especially those between the ages of 30 and 50 and those of European ethnicity (50% accuracy). On the other hand, the model has perfect accuracy for female applicants.

Table 1: A testing set for a loan approval model.

id	sex	age	ethnicity	income	label	prediction
1	M	<30	Hispanic	90	0	0
2	M	<30	Black	35	0	0
3	M	30-50	Hispanic	74	0	0
4	M	30-50	European	70	1	0
5	M	30-50	European	39	0	0
6	M	>50	European	50	1	0
7	M	>50	European	36	0	0
8	M	30-50	Hispanic	210	0	1
9	M	>50	Black	100	1	1
10	F	<30	European	15	0	0
11	F	<30	Hispanic	50	0	0
12	F	<30	Black	85	0	0
13	F	30-50	Hispanic	73	0	0
14	F	>50	Black	46	0	0
15	F	>50	Hispanic	20	0	0
16	F	30-50	European	150	1	1

Table 2: Slicing a loan approval model.

sex	age	ethnicity	accuracy	support
M	30-50	*	50	4
M	*	European	50	4
M	*	*	67	9
*	30-50	*	67	6
*	*	European	67	6
*	>50	*	80	5
*	*	*	81	16
*	*	Hispanic	83	6
F	*	*	100	7
*	<30	*	100	5
*	*	Black	100	4

Model slicing insights can improve data quality and mitigate model bias. For example, a model may underperform for a given subgroup if there are systematic quality issues in the training data related to this subgroup, or if there is insufficient training data for this subgroup [2]. Furthermore, expected model performance at inference time can be estimated using model slicing [6, 17]. In the above example, financial experts may want to examine loan approval results of male applicants knowing that the model has low accuracy for this subgroup.

The purpose of this article is to review recent research on model slicing for responsible AI. Toward this goal, we propose a general solution template for model slicing, we situate recent work in this space (Section 2), and we conclude with a list of open research directions (Section 3).

2 A SURVEY OF MODEL SLICING

We categorize model slicing along the following dimensions:

- the slicing (or grouping) attributes,
- the slicing metric (e.g., model accuracy),
- the set (or sorted list) of groups displayed,
- whether one model or several models are tested (e.g., to find subgroups where models make different predictions),
- and whether the dataset being sliced is the testing set (e.g., to identify underperforming subgroups, as in Table 2) or the training set (e.g., to identify training examples that contribute to model bias).

We represent variants of model slicing in pseudo-SQL syntax, shown below. Figure 1 illustrates the types of model slicing problems that will be reviewed in this section, and Table 3 lists the methods surveyed and their functionality, as described in the corresponding papers. From now on, we will use the terms subgroup and slice interchangeably.

```
SELECT <slicing attributes>, AGGR(<slicing metric>)
FROM <training set / testing set>
GROUP BY CUBE(<slicing attributes>)
HAVING <slicing condition>
ORDER BY <sorting condition>
```

[itemize]noitemsep,nolistsep

slicing attributes
- model features
- discovered via supervised learning
- discovered via unsupervised learning

slicing dataset
- training
- testing

slicing metrics
- prediction bias
  - *model accuracy*
  - *model loss*
- recourse bias
  - *recourse availability*
  - *recourse cost*
  - *recourse options*
- *additionally for multi-model slicing*
  - *consensus*
  - *logic*

slicing conditions
- support
- significance

sorting conditions
- slicing metric
- informativeness

Figure 1: Variants of model slicing.

2.1 Slicing Attributes

Slicing attributes define the subgroups for which model performance metrics will be computed. For models learning from tabular data, slicing attributes correspond to some or all of the model features. For models learning from unstructured data such as images, model slicing requires an additional preliminary step of *discovering* the slicing attributes. We distinguish between supervised and unsupervised slicing attribute discovery.

Supervised approaches annotate testing examples with structured slicing attributes. Suppose that we want to slice a neural network for classifying images of indoor locations into kitchens, living rooms and bedrooms [4, 6]. We may annotate each image with the objects present in it such as tables, chairs, sofas and beds. These annotations become binary slicing attributes, denoting the presence or absence of objects in images. If chairs exist in images of various types of locations, then model slicing might find that the model has trouble classifying images with chairs, i.e., it underperforms for the subgroup defined as "chairs=1".

Unsupervised slicing attribute discovery methods cluster the testing set and assign interpretable labels to each slice. *Domino* is one such example [7], which clusters images based on their embeddings and generates textual cluster captions using a cross-modal embedding model such as *CLIP* [15].

2.2 Slicing Output

Early work, *MLCube*, produced the full data cube of slices [10]. Recent work has proposed various slicing conditions (corresponding to the HAVING clause) and sort orders (corresponding to ORDER BY). Three common slicing conditions are as follows.

- Displaying slices that exceed a given support threshold, used in *SliceFinder* [3] and *SliceLine* [16] (recall Table 2).
- Displaying slices that exceed a given statistical significance threshold, used in *SliceFinder* [3].
- Displaying slices that summarize the model's accuracy distribution, used in *InfoMoD* [6], which selects the most 'informative' slices that over or under perform the average.

The total number of candidate slices is equal to the size of the data cube over the slicing attributes. Many model slicing solutions include scalability optimizations that exploit the structure of the problem variant at hand. For example, both *SliceFinder* and *SliceLine* use a minimum support threshold in the HAVING clause and therefore can employ Apriori-style optimizations from frequent itemset mining to ignore slices failing the support threshold.

2.3 Slicing Metrics

We distinguish between metrics targeting *prediction* and *recourse* bias. Prediction bias exists when a model performs worse on some slices than others. As seen in Table 2, one metric targeting this type of bias is the fraction of accurate predictions in each slice. Model loss (e.g,. quadratic or cross-entropy loss [5]) is another option. For example, a model that predicts the right class with 90 percent probability would have a lower loss than one that predicts the right class with 75 percent probability.

Table 3: A summary of the surveyed model slicing methods.

method	attributes	data	multi-model?	metrics	conditions
CAMO [20]	features	testing	yes	accuracy, consensus, logic	informativeness
datamodels [9]	discovered	training	no	loss	top-k
Domino [7]	discovered	testing	no	accuracy	top-k
FACTS [12]	features	testing	no	recourse cost	top-k
FUME [19]	features	training	no	loss	top-k
GOPHER [14]	features	training	no	loss	top-k
InfoMoD [6]	features, discovered	testing	no	accuracy, false positive/negative rate	informativeness
MLCube [10]	features	testing	yes	accuracy, loss	full cube
ModelDiff [18]	discovered	training	yes	loss	top-k
REACT [1]	features	testing	no	recourse availability, cost, options	informativeness
SliceFinder [3]	features	testing	yes	accuracy, loss	support, significance
SliceLine [16]	features	testing	no	loss	support, top-k

Furthermore, a model may be biased if it is harder for some subgroups to achieve *recourse*, i.e., to flip the model's undesirable prediction (e.g., loan denied) to a desirable one [8]. For example, what if married individuals whose loan applications were rejected would only need to increase their incomes by an average of ten percent to be approved, but single individuals would need 20 percent higher salaries?

Let R be a subset of attributes that are selected be modified to achieve recourse. For example, perhaps salary can be modified, but not age. Furthermore, the allowed changes can be bounded, e.g., salary can only increase by 20 percent [11]. The following slicing metrics have been proposed for recourse bias.

- **Recourse availability**: one if there is any way to modify the attributes in R in a way that flips the model's decision, zero otherwise.
- **Recourse cost**: if recourse is possible, what is the recourse cost or recourse burden, in terms of the magnitude of change to the attributes in R?
- **Recourse options**: how many different paths to recourse are there? For example, perhaps a denied loan would have been accepted if salary were higher or if the applicant had less credit card debt, leading to two recourse options.

Recall Table 1 and assume that $R = \{income\}$. Take the declined applicants (with prediction=0) and suppose that the minimal income increases that would lead to the model approving their loans are as shown in Table 4 under recourse cost, with the other attributes copied from Table 1 (example adapted from [20]). For instance, the first example (id=1) would have to increase its income from 90 to 100. Here, all declined applicants happen to have recourse availability, but this may not always be the case.

Let us use sex, age and ethnicity as the slicing attributes and recourse cost as the slicing metric. Table 5 shows the slices with support at least four, sorted in descending order of recourse cost. We see that young applicants (age < 30), female applicants, and Hispanic applicants would pay a higher recourse cost.

FACTS [12] and *REACT* [1] are two solutions for recourse bias slicing. *FACTS* uses recourse cost as the slicing metric and outputs the top-k slices with the highest cost. *REACT* supports recourse cost, availability and options, and, similar to *InfoMoD*, outputs the most informative slices with a substantially higher or lower than average slicing metric. Note that slicing can also be done by exploring the cost of flipping a model's decision in the other direction, from favourable to unfavourable.

Table 4: Recourse costs of the declined examples from Table 2.

id	sex	age	ethnicity	income	recourse cost
1	M	<30	Hispanic	90	10
2	M	<30	Black	35	70
3	M	30-50	Hispanic	74	16
4	M	30-50	European	70	8
5	M	30-50	European	39	31
6	M	>50	European	50	15
7	M	>50	European	36	24
10	F	<30	European	15	80
11	F	<30	Hispanic	50	60
12	F	<30	Black	85	18
13	F	30-50	Hispanic	73	25
14	F	>50	Black	46	20
15	F	>50	Hispanic	20	68

Table 5: Slicing a loan approval model by recourse cost.

sex	age	ethnicity	cost	support
*	<30	*	47.6	5
F	*	*	45.2	6
*	*	Hispanic	35.8	5
*	*	*	34.2	13
*	>50	*	31.8	4
*	*	European	31.6	5
M	*	*	24.9	7
*	30-50	*	20	4
M	*	European	19.5	4

2.4 Multi-Model Slicing

The following multi-model slicing metrics have been explored in the literature (assume that the same testing set is used by all models).

- **Consensus:** one if all models make the same decision for a given testing example, and zero otherwise.

- **Expertise:** the difference between the accuracy or loss of one model versus others, to find slices where that model performs better or worse than others.
- **Decision-making Logic:** one if all models pay attention to the same features when making a prediction for a given example, and zero otherwise. One way to quantify model attention is to assign an influence score to each feature using methods such as Shapley values [13].

MLCube, *SliceFinder* and *CAMO* [20] are designed for or at least mention multi-model slicing. *MLCube* and *SliceFinder* focus on single-model slicing using model accuracy or loss as the metric, with a simple extension to two models by taking the difference of their accuracies or losses for each testing example. *CAMO*, designed specifically for model comparisons, supports consensus, expertise and decision-making logic, using the same slicing condition as *InfoMoD* and *REACT* (select the most informative slices).

Our categorization in Figure 1 also includes recourse bias metrics, suggesting a gap in the literature: multi-model slicing to detect recourse bias. Though not implemented in *CAMO* or any other existing methods (to best of our knowledge), the expertise metric can be redefined to assess recourse bias. For example, we can calculate the difference between the recourse cost of one model versus others to find slices where that model causes a high recourse burden.

2.5 Training Set Slicing

Take a testing set a metric, e.g., model accuracy or loss on the testing set. Next, "unlearn" a slice of the training set, i.e., either retrain the model with that slice removed or approximate such a model without explicitly retraining it using methods such as *datamodels* [9]. The accuracy or loss difference (on the testing set) between the full model and the retrained model becomes the slicing metric for the unlearned slice of the training set. As a result, we can identify training set slices that reduce the accuracy of the model, perhaps due to errors in those slices such as incorrect labels. Similarly, we can find slices that contribute positively to model accuracy, with implications for data acquisition and data valuation markets. For example, given a testing set of images, *datamodels* identifies clusters of positively and negatively contributing training images.

FUME [19] and *GOPHER* [14] perform training set slicing to detect prediction bias with respect to a particular protected attribute. First, separately compute full model accuracy or loss on the testing set for different values of the protected attribute, e.g., male vs. female. Let d be the difference in model accuracy or loss between the values of the protected attribute. Then, for each candidate slice, unlearn the slice and recompute model accuracy or loss on the testing set for different values of the protected attribute. Let d' be the new difference in model accuracy or loss between the values of the protected attribute, corresponding to the partial model trained without the candidate slice. The slicing metric for this slice is $d - d'$, i.e., the contribution of that slice to model bias, or how much bias (with respect of the protected attribute) can be taken away by removing that slice from the training set.

As an example, similar to [14], suppose that we slice a model that predicts whether an individual makes a high salary (say, over $\$50,000$) based on features such as sex and occupation. Suppose that there is a systematic error in the training set such that federal government workers are listed with their total household income, not their individual income, inflating their reported salaries. Suppose further that the federal government workers in the training set are mostly male. Given this data quality issue, the model would most likely be biased. That is, it would be more likely to predict a higher salary for males than females. However, this bias would be reduced if we unlearned the training set slice corresponding to "sex = M and occupation = federal government".

Both *FUME* and *GOPHER* output the top-k training data slices with the greatest contribution to bias. Notably, the approach taken by *FUME* and *GOPHER* is to compute the slicing metric for each training slice directly instead of aggregating it from the metrics of individual training examples, as in our SQL template. This is more efficient when combined with a minimum support threshold, which enables Apriori-like optimizations from frequent itemset mining, similar to those used by *SliceFinder* and *SliceLine* for testing set slicing. As another optimization, *FUME* and *GOPHER* do not re-train models on training subsets with some slices removed, but rather they focus on different types of models for which the effect of unlearning can be efficiently estimated.

Training set slicing can be extended to multiple models. For example, *ModelDiff* [18] extends *datamodels* by identifying training examples that, when unlearned, increase or drop the testing set accuracy of one model but not the other. And, though not considered by *FUME* or *GOPHER*, these tools can also be extended to find training set slices that, when unlearned, reduce the bias of one model but not the other.

The training set slicing methods described in this section focus on prediction bias. Returning to our slicing design space in Figure 1, there appears to be another gap in the literature: training set slicing for recourse bias. Though not considered explicitly by *FUME* or *GOPHER*, these two approaches can be extended to recourse bias in a straightforward way. Instead of measuring the reduction in accuracy due to unlearning a slice, we can measure the reduction in recourse cost difference between the values of the protected attribute due to unlearning a slice. For example, we can find training slices whose unlearning equalizes the recourse cost for males and females whose loan applications were declined.

3 OPEN PROBLEMS

We conclude with the following directions to improve the efficiency and effectiveness of model slicing.

- **Slicing Large Language Models (LLMs).** Existing work focuses on models for structured and image inputs. Slicing LLMs is an interesting direction for future work that introduces new challenges such as formulating the slicing attributes for different types of generative tasks
- **Scalability.** As discussed in Section 2.2, various strategies can be used to prune the space of candidate slices, especially with slicing conditions in the HAVING clause such as minimum support. However, slicing complexity of the extended formulations, i.e., those for recourse bias slicing or training set slicing, is higher. In recourse bias slicing, searching for recourse paths is the bottleneck, which requires probing the model repeatedly with perturbed features to check if it

makes a different prediction. New optimizations are needed to prune this search space.

- **Integration with AI pipelines.** Existing efforts focus on individual model slicing problems, such as finding statistically significant slices or summarizing recourse bias. These individual solutions could be generalized to form an interactive slice-based AI debugging tool, allowing users to slice datasets in different ways, and enabling user-guided exploration that prioritizes slices based on user requirements.

REFERENCES

[1] Anastasiia Avksientieva, Parke Godfrey, Lukasz Golab, Divesh Srivastava, and Jarek Szlichta. 2025. REACT: REcourse Analysis with Counterfactuals and Explanation Tables. In *Proceedings 28th International Conference on Extending Database Technology, EDBT*. 1098–1101. https://doi.org/10.48786/EDBT.2025.98

[2] Jiwon Chang, Christina Dionysio, Fatemeh Nargesian, and Matthias Boehm. 2024. PLUTUS: Understanding Data Distribution Tailoring for Machine Learning. In *Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS*, Pablo Barceló, Nayat Sánchez-Pi, Alexandra Meliou, and S. Sudarshan (Eds.). 528–531. https://doi.org/10.1145/3626246.3654745

[3] Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. 2020. Automated Data Slicing for Model Validation: A Big Data - AI Integration Approach. *IEEE Trans. Knowl. Data Eng.* 32, 12 (2020), 2284–2296. https://doi.org/10.1109/TKDE.2019.2916074

[4] Vargha Dadvar, Lukasz Golab, and Divesh Srivastava. 2023. POEM: Pattern-Oriented Explanations of Convolutional Neural Networks. *Proc. VLDB Endow.* 16, 11 (2023), 3192–3200. https://doi.org/10.14778/3611479.3611518

[5] Ahmet Demirkaya, Jiasi Chen, and Samet Oymak. 2020. Exploring the Role of Loss Functions in Multiclass Classification. In *54th Annual Conference on Information Sciences and Systems, CISS*. 1–5. https://doi.org/10.1109/CISS48834.2020.1570627167

[6] Armin Esmaelizadeh, Sunil Cotterill, Liam Hebert, Lukasz Golab, and Kazem Taghva. 2025. Infomod: information-theoretic machine learning model diagnostics. *Distributed Parallel Databases* 43, 1 (2025), 6. https://doi.org/10.1007/S10619-024-07450-8

[7] Sabri Eyuboglu, Maya Varma, Khaled Kamal Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. 2022. Domino: Discovering Systematic Errors with Cross-Modal Embeddings. In *The Tenth International Conference on Learning Representations, ICLR*.

[8] Christos Fragkathoulas, Vasiliki Papanikou, Danae Pla Karidi, and Evaggelia Pitoura. 2024. On Explaining Unfairness: An Overview. In *40th International Conference on Data Engineering, ICDE 2024 - Workshops*. 226–236. https://doi.org/10.1109/ICDEW61823.2024.00035

[9] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. 2022. Datamodels: Understanding Predictions with Data and Data with Predictions. In *International Conference on Machine Learning, ICML*, Vol. 162. 9525–9587.

[10] Minsuk Kahng, Dezhi Fang, and Duen Horng (Polo) Chau. 2016. Visual exploration of machine learning results using data cube analysis. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. Article 1, 6 pages. https://doi.org/10.1145/2939502.2939503

[11] Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. 2023. A Survey of Algorithmic Recourse: Contrastive Explanations and Consequential Recommendations. *ACM Comput. Surv.* 55, 5 (2023), 95:1–95:29. https://doi.org/10.1145/3527848

[12] Loukas Kavouras, Konstantinos Tsopelas, Giorgos Giannopoulos, Dimitris Sacharidis, Eleni Psaroudaki, Nikolaos Theologitis, Dimitrios Rontogiannis, Dimitris Fotakis, and Ioannis Z. Emiris. 2023. Fairness Aware Counterfactuals for Subgroups. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems*.

[13] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*. 4765–4774.

[14] Romila Pradhan, Jiongli Zhu, Boris Glavic, and Babak Salimi. 2022. Interpretable Data-Based Explanations for Fairness Debugging. In *SIGMOD '22: International Conference on Management of Data*. 247–261. https://doi.org/10.1145/3514221.3517886

[15] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, Vol. 139. 8748–8763.

[16] Svetlana Sagadeeva and Matthias Boehm. 2021. SliceLine: Fast, Linear-Algebra-based Slice Finding for ML Model Debugging. In *SIGMOD '21: International Conference on Management of Data*. 2290–2299. https://doi.org/10.1145/3448016.3457323

[17] Sebastian Schelter, Tammo Rukat, and Felix Bießmann. 2020. Learning to Validate the Predictions of Black Box Classifiers on Unseen Data. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference*. 1289–1299. https://doi.org/10.1145/3318464.3380604

[18] Harshay Shah, Sung Min Park, Andrew Ilyas, and Aleksander Madry. 2023. ModelDiff: A Framework for Comparing Learning Algorithms. In *International Conference on Machine Learning, ICML*, Vol. 202. 30646–30688.

[19] Tanmay Surve and Romila Pradhan. 2025. Explaining Fairness Violations using Machine Unlearning. In *Proceedings 28th International Conference on Extending Database Technology, EDBT*. 623–635. https://doi.org/10.48786/EDBT.2025.50

[20] Andy Yu, Parke Godfrey, Lukasz Golab, Divesh Srivastava, and Jaroslaw Szlichta. 2024. CAMO: Explaining Consensus Across MOdels. In *40th IEEE International Conference on Data Engineering, ICDE*. 5493–5496. https://doi.org/10.1109/ICDE60146.2024.00436