Appendix: SYNG4ME: Model Evaluation using Synthetic Test Data

Table of Contents

A EXTENDED RELATED WORK

We present a comparison of our framework, SYNG4ME, and provide further contrast to related work. Table 2, highlights that related methods often do not permit automated test creation—in particular, they are often human labor-intensive. Benchmark tasks are tailored to specific datasets and tasks and hence cannot be customized and/or personalized to an end-user's specific task, dataset or trained model. Finally, both benchmark methods and behavioral testing require additional data to be collected or created—in contrast to SYNG4ME which only requires the original test data, already available.

Table 2: Comparison of related work. (i) Automated test creation, (ii) Permits use case test personalization and (iii) No additional data or info required.

	Approach	Assumption	Use cases	(i)	(ii)	(iii)
SYNG4ME (Ours)	Synthetic test data	Generative model fits the data	-Subgroup testing -Distributional shift testing	✓	✓	✓
BENCHMARK TASKS						
Imagenet-C/P (Hendrycks & Dietterich, 2018)	Create corrupted images	Synthetic corruption reflects the real-world	Image corruption testing	×	×	×
Wilds (Koh et al., 2021)	Collect data with real shifts	Wild datasets reflect sufficient use cases	Distribution shift testing	×	×	×
MODEL BEHAVIOURAL TESTING						
CheckList (Ribeiro et al., 2020)	Human crafted test scenarios	Know a-priori scenarios to test	Crafted test scenario tests	×	✓	×
HateCheck (Röttger et al., 2021)	Human crafted test scenarios	Know a-priori scenarios to test	Crafted test scenario tests	×	✓	×
AdaTest (Ribeiro & Lundberg, 2022)	GPT-3 creates tests, human selects	Human-in-the-loop	Weakness probing	✓	✓	✓

Data-centric AI. The usage and assessment is vital for ML models, yet is often overlooked as operational (Sambasivan et al., 2021). The recent focus on data-centric AI (Ng et al., 2021) aims to build systematic tools to improve the quality of data used to train and evaluate ML models (Polyzotis & Zaharia, 2021). SYNG4ME contributes to this nascent body of work, specifically around the usage and generation of data to better evaluate and test ML models

Simulators and counterfactual generators. Simulators have been used to benchmark algorithms in settings such causal effect estimation and sequential decision-making (Chan et al., 2021). The work on simulators is not directly related, as the goal is often to test scenarios that are not available in real-world data. For example, in (Chan et al., 2021) the goal is customization of the decision-making policy in order to evaluate methods to better understand human decision-making. Similarly, there has been work on generating realistic synthetic data in the causal inference domain (Neal et al., 2020; Athey et al., 2021; Parikh et al., 2022), but these methods focus on benchmarking causal inference methods, do not consider distributional shifts nor subgroup evaluation, and do not explore the value of the synthetic data beyond realistic, ground-truth counterfactuals.

Synthetic data in computer vision. Synthetic data has been used in computer vision both to improve model training and to test weaknesses in models. These methods can be grouped as follows by their motivations:

- Generate synthetic data for training, to reduce the reliance on collecting and annotating large training sets— This is different from SYNG4ME as they focus on constructing better training sets, rather than constructing better test sets for model evaluation
- Generate synthetic data to improve the model by augmenting the real dataset with synthetic examples—*Again, this is different from SYNG4ME as they focus on training better-performing models, rather than the evaluation of an already trained model
- Generate synthetic data to probe models on different dataset attributes. For example, in face recognition, how the model might perform on faces with long vs short hair. This is most similar to SYNG4ME, but there are clear differences in both the goal for and approach to generating the data. We compare SYNG4ME to (i) CGI- or physics-based simulators and (ii) deep generative models for probing in computer vision.

In Table 3, we contrast both simulators and generative approaches where synthetic data to probe models on different dataset attributes.

Table 3: Comparing SYNG4ME to computer vision synthetic data approaches. A is a performance metric (Eq. 1) and f the trained predictive model. $\mathcal{D}_{syn}(x)$ denotes synthetic data is created dependent on x

Examples	Data and/or generator input	Conditioning info	Does not require pretrained simulator/generator	$\mathcal{D}_{syn}$ used for training or testing	Goal
SYNG4ME - Subgroup	$\mathcal{D}_{train,G} = \mathcal{D}_{test,f}$ (Any dataset)	Subgroups $\mathcal{S}$	✓	Testing	Reliable subgroup performance estimates for f, i.e. choose $\mathcal{D}_{syn} \sim p_G$ s.t. $A(f;\mathcal{D}_{syn},\mathcal{S})] \approx \mathbb{E}_{\mathcal{D}^{iid} \sim p_{\mathcal{S}}}[A(f;\mathcal{D},\mathcal{S})]$
SYNG4ME - Shift		Shift information T			Estimate performance of f under shift T, i.e. choose $\mathcal{D}_{syn} \sim p_G$ s.t. $A(f;\mathcal{D}_{syn},\mathcal{S})] \approx \mathbb{E}_{\mathcal{D}^{iid} \sim p^T}[A(f;\mathcal{D},\mathcal{S})]$
Computer Vision					
Wang et al. (2019)	Video game engine (GTA5) Real-world data	Scene info $\mathcal{S}$ in virtual world	×	Training	Improve crowd counting performance on diff. scenes by generating semi-synthetic data for training f, i.e. $\max_f A(f;\mathcal{D}_{test}(\mathcal{S}))$
Trigueros et al. (2021)	$\mathcal{D}_{train,f}$ = VGGFace	Identity attributes	✓	Training	Improve overall performance of facial recognition i.e. $\max(A(f;\mathcal{D}_{test}))$
Kortylewski et al. (2019)	3D face model	Nuisance transforms $\mathcal{N}$	×	Testing	Report face recognition robustness to different nuisances $\mathcal{N}$, $\mathcal{D}_{syn}(N)$ and report $A(f;\mathcal{D}_{syn}(N)), \forall N \in \mathcal{N}$
Ruiz et al. (2022)	3D face model	Simulator parameters ρ	×	Testing	Find adversarial failures for face recognition, i.e. find $\rho = \arg\min_\rho A(f;D_{syn}(\rho))$
Khan et al. (2019)	CityEngine, Unreal Engine, CARLA	Weather conditions $\mathcal{S}$	×	Testing	Report segmentation performance for self-driving cars under different weather conditions, i.e. $\mathcal{D}_{syn}(S)$ and report $A(f;\mathcal{D}_{syn}(S)), \forall S \in \mathcal{S}$
Li & Xu (2021)	Pretrained StyleGAN	Implicit attributes S (e.g. age, lighting)	×	Testing	Find attributes with poor performance, i.e. $\text{argmin}_S(A(f;\mathcal{D}_{syn}(S)))$
McDuff et al. (2019)	$\mathcal{D}$ = MS-CELEB-1M	Subgroups S	✓	Testing	Find S with poor face recognition performance, i.e. $\text{argmin}_S(A(f;\mathcal{D}_{syn}(S))$

Synthetic data and tabular approaches.

Thank you for highlighting these two works (DataSynthesizer and AITEST). We contrast SYNG4ME to two works DataSynthesizer (Howe et al., 2017) and AITEST (Saha et al., 2022), which while seemingly similar have specific differences to SYNG4ME. A side-by-side contrast is presented in Table 4.

Data Synthesizer

We believe SYNG4ME is significantly different from DataSynthesizer, in terms of aims, assumptions and algorithmically.

Aim and assumptions. Data Synthesizer primarily focuses on privacy preserving generation of tabular synthetic data. The closest component to our work is the extension the paper proposes around adversarial fake data generation. While there are no experiments, the adversarial fake data consists of three areas. We contrast them to SYNG4ME.

The major difference is Data Synthesizer assumes access to **full knowledge about the shift/distributional change**. In contrast, SYNG4ME operates in a different setting - (1) **No prior knowledge** on the shift and (2) **high-level partial knowledge** about the shift through observing some variables in the target domain.

1. Edit the distribution: this assumes the user knows exactly the shift [Full knowledge of the shift]. SYNG4ME covers two different settings: (1) No prior knowledge on the shift, where only minimal assumptions on means of variables allow us to create characteristic curves like in Section 5.2.* and (2) Incorporating prior knowledge, in which some features are observed from the shifted distribution and we use these to generate the full data from the shifted distribution, like in Section 5.2.2. Consequently, the difference is that SYNG4ME tackles the no and partial information settings, whereas Data Synthesizer tackles the full info setting of editing the distribution.
2. Preconfigured pathological distributions — this requires full and exact knowledge about the shift, which differs from SYNG4ME of partial knowledge and no prior knowledge settings.
3. Injecting missing data, extreme values — either such an approach is possible to incorporate in SYNG4ME. We see these ideas as complementary.

Algorithmic.

The authors propose three methods, one with random features, one with independent features and one with correlated features. Due to the absence of correlation in the first two, these reduce the data utility. Let us thus focus on the third method, that does include correlation. This approach uses Bayesian Networks and is only applicable to discrete data, hence needing to discretise continuous variables. This loses utility when a coarse discretisation is chosen, while a fine discretisation is often intractable and data-inefficient due to the ordinal information being lost, e.g. results for $age = 31$ and $age = 32$ will generally be similar—exactly the reason why the independent approach was also introduced. Bayesian Nets are also limited in other ways, e.g. results can be influenced by the feature generation order deviating from the real data generation process' ordering, as indicated by the authors of DataSynthesizer in Figures 5 and 6.

AITEST

We will include AITEST in the related work in the *updated Appendix A*, since indeed it uses synthetic data for testing. Let us briefly outline why we believe SYNG4ME is novel and significantly different compared to AITEST. We contrast SYNG4ME to AITEST in terms of aims and assumptions, algorithm, and use cases.

Aims and assumptions. AITEST has a significantly different aim and method compared to SYNG4ME. As mentioned by the reviewer, AITEST can test for adversarial robustness by generating realistic data with user-defined constraints, but this is different from our work that aims to generate synthetic test data for granular evaluation and distributional shifts.

Additionally, the assumptions on user input are quite different: AITEST enables users to define constraints on features and associations between features, whereas SYNG4ME requires information in terms of which subgroups to test or shifts to generate. We do see possibilities to combine

both frameworks, e.g. through including constraints similar to the ones AITEST uses within the SYNG4ME method, or using fairness as a downstream task.

We have taken a step in this direction and added fairness as an additional experiment and have included this experiment in the new Appendix D.4.

Algorithmic AITEST requires a decision tree surrogate of the black-box model, whilst SYNG4ME does not need to model the black-box predictive model. AITEST defines data constraints by fitting different distributions to the features and using statistical testing to select the correct distribution. The dependencies are then captured by a DAG. SYNG4ME does not require predefined constraints and dependencies, but aims to learn these implicitly with the generative model.

Use cases

- Group fairness: AITEST aims to probe if a model does have a group fairness issue or not. The goal of SYNG4ME is different — even if models don't have group bias issues, with SYNG4ME we desire reliable performance metric estimates (accuracy or even fairness) which are similar to the oracle estimates on small and intersectional subgroups for which we have limited real test data.
- Adversarial robustness: AITEST does this by generating more inputs in the neighborhood of a specific sample and seeing if they behave the same. In reality, this is analogous to group-wise testing with $n = 1$, a very specific type of group testing. In contrast, with SYNG4ME we explore multiple definitions of groups from specific sensitive attributes, to intersectional groups, to points of interest (i.e. $n = 1$), to high- and low density regions.
- AITEST does not account for distribution shift, unlike SYNG4ME which looks at distribution shift with no prior knowledge and high-level knowledge.

Table 4: Comparing SYNG4ME to other tabular approaches of generating synthetic test data. f is the trained predictive model and we abbreviate $A(f; \mathcal{D}) = A(f; \mathcal{D}, \Omega)$ for evaluating f over all of $\mathcal{D}$ (Eq. 1). (i) used for evaluating subgroups, (ii) used for evaluating shifts, (iii) does not require discretisation of continuous features, (iv) does not require modelling black-box f

Examples	Inputs	(i)	(ii)	(iii)	(iv)	Generator Type	Goal
SYNG4ME	$\mathcal{D}_{train,G} = \mathcal{D}_{test,f}$ (Any dataset) Subgroups: S, Shifts: No/Partial knowledge	✓	✓	✓	✓	GAN	Reliable subgroup performance estimates for f, i.e. choose $\mathcal{D}_{syn} \sim p_G$ s.t. $A(f; \mathcal{D}_{syn}, \mathcal{S})] \approx \mathbb{E}_{\mathcal{D} \sim p_R}[A(f; \mathcal{D}, \mathcal{S})]$ Estimate performance of f under shift T, i.e. choose $\mathcal{D}_{syn} \sim p_G$ s.t. $A(f; \mathcal{D}_{syn}, \mathcal{S})] \approx \mathbb{E}_{\mathcal{D} \sim p^s}[A(f; \mathcal{D}, \mathcal{S})]$
DataSynthesizer Howe et al. (2017)	Privacy-sensitive $\mathcal{D}$ Full knowledge of shift	×	✓	×	✓	Bayesian network	Generate private data, extensions for generating pathological data through (i) manual editing, (ii) inserting extreme values/missingness, and (iii) .
AITEST Saha et al. (2022)	$\mathcal{D}_{train,G} = \mathcal{D}_{train,f}$, Constraints and dependencies (DAG)	✓	×	×/✓	×	Sample features sequentially following DAG	Subgroup performance, e.g. fairness between two sensitive groups (S_1, S_2), i.e. $\frac{A(f; \mathcal{D}_{Syn}, S_1)}{A(f; \mathcal{D}_{Syn}, S_2)}$

B EXPERIMENTAL DETAILS

This appendix includes details on the experiments, including (i) the datasets, and (ii) the different settings of the experiments, including the implementation of baselines.

B.1 DATASETS

Here we describe the real-world datasets used in greater detail.

ADULT Dataset The ADULT dataset (Asuncion & Newman, 2007) has 32,561 instances with a total of 13 attributes capturing demographic (age, gender, race), personal (marital status) and financial (income) features amongst others. The classification task predicts whether a person earns over $50K or not. We encode the features (e.g. race, sex, gender etc) and a summary can be found in Table 5.

Note that there is an imbalance across certain features, such as across different race groups, which is what we evaluate.

Table 5: Summary of features for the ADULT Dataset (Asuncion & Newman, 2007)

Feature	Values/Range
Age	$17-90$
education-num	$1-16$
marital-status	$0,1$
relationship	$0,1,2,3,4$
race	$0,1,2,3,4$
sex	$0,1$
capital-gain	$0,1$
capital-loss	$0,1$
hours-per-week	$1-99$
country	$0,1$
employment-type	$0,1,2,3$
salary	$0,1$

Covid-19 Dataset The Covid-19 dataset (Baqui et al., 2020) consists of Covid patients from Brazil. The dataset is publicly available and based on SIVEP-Gripe data (Brazil Ministry of Health). The dataset consists of 6882 patients from Brazil recorded between February 27-May 4 2020. The dataset captures risk factors including comorbidities, symptoms, and demographic characteristics. There is a mortality label from Covid-19 making it a binary classification task. A summary of the characteristics of the covariates can be found in Table 6.

SEER Dataset The SEER dataset is a publicly available dataset consisting of 240,486 patients enrolled in the American SEER program (Duggan et al., 2016). The dataset consists of features used to characterize prostate cancer, including age, PSA (severity score), Gleason score, clinical stage, and treatments. A summary of the covariates can be found in Table 7. The classification task is to predict patient mortality, which is a binary label.

The dataset is highly imbalanced, where 94% of patients survive. Hence, we extract a balanced subset of of 20,000 patients (i.e. 10,000 with label=0 and 10,000 with label=1).

CUTRACT Dataset The CUTRACT dataset is a private dataset consisting of 10,086 patients enrolled in the British Prostate Cancer UK program (Prostate Cancer UK). It includes the same features as SEER and also uses mortality as label, see Table 7.

The dataset is highly imbalanced in its labels, hence we choose to extract a balanced subset of 2,000 patients (i.e. 1000 with label=0 and 1000 with label=1).

B.2 EXPERIMENTS

For specifics on how G is evaluated, tuned and selected, please see Appendix C.1.1.

Table 6: Summary of features for the Covid-19 Dataset (Baqui et al., 2020)

Feature	Range
Sex	0 (Female), 1(Male)
Age	$1-104$
Fever	$0,1$
Cough	$0,1$
Sore throat	$0,1$
Shortness of breath	$0,1$
Respiratory discomfort	$0,1$
SPO2	$0-1$
Diharea	$0,1$
Vomitting	$0,1$
Cardiovascular	$0,1$
Asthma	$0,1$
Diabetes	$0,1$
Pulmonary	$0,1$
Immunosuppresion	$0,1$
Obesity	$0,1$
Liver	$0,1$
Neurologic	$0,1$
Branca (Region)	$0,1$
Preta (Region)	$0,1$
Amarela (Region)	$0,1$
Parda (Region)	$0,1$
Indigena (Region)	$0,1$

Table 7: Summary of features for the SEER (Duggan et al., 2016) and CUTRACT Prostate Cancer UK datasets. *Note: the range of age starts slightly lower for SEER (37-95) compared to CUTRACT (44-95).*

Feature	Range
Age	$37-95$
PSA	$0-98$
Comorbidities	$0,1,2,\geq 3$
Treatment	Hormone Therapy (PHT), Radical Therapy - RDx (RT-RDx),Radical Therapy -Sx (RT-Sx), CM
Grade	$1,2,3,4,5$
Stage	$1,2,3,4$
Primary Gleason	$1,2,3,4,5$
Secondary Gleason	$1,2,3,4,5$

B.2.1 Experiment 5.1: Subgroups

In this experiment, we evaluate the performance estimates on different subgroups based on the mean absolute error compared to the estimates of subgroup performance using the oracle dataset. In order, to represent potential variation of selecting different test sets, we repeat the experiment 10 times, where we sample a different test set in each run. That being said, we keep the proportions in each dataset fixed such that $\{\mathcal{D}_{train,f}$,$\mathcal{D}_{test,f}$,$\mathcal{D}_{oracle}\}$ = $\{8.4k, 2.1k, 19.6k\}$. Given that minority subgroups have few samples, in this experiment, we generate n samples for each subgroup, where n is the size of the largest subgroup in $\mathcal{D}_{test,f}$. This allows us to "balance" the evaluation dataset.

In producing the intersectional performance matrix, we slice the data for these intersections. However, as we slice the data into finer intersections, the intersectional groups naturally become smaller. Hence, to ensure we have reliable estimates, we set a cut-off wherein we only evaluate performance for intersectional groups where there are 100 or more samples. In computing the mean absolute error, we do not include the corresponding intersections for which there were insufficient samples.

B.2.2 Experiments 5.2 Distributional shifts

No prior knowledge: characterizing sensitivity across operating ranges. In this experiment, we shift the marginal distribution of single features in mean; i.e. we can write for the shifted marginal $p^s(X_i) = p^0(X_i - s)$. We vary s over $\pm\sigma(X_i)$, with $\sigma(X_i)$ the empirical standard deviation of X_i. To mitigate unrealistic values, we use cut-offs on both ends of the marginal distribution, i.e. we

discard any samples that fall outside the original range of X_i. We repeat the sampling from G over 5 independent shifts with the same value of s, in order to represent the potential variability of the estimates. The Oracle target is created using rejection sampling of the oracle source data, see Section B.3.

Prior Knowledge. In this experiment, we assume we observe some of the feature in the target domain, i.e. we observe the empirical marginal distribution of X_c. This empirical marginal distribution is used to sample from, and conditioned on when generating the other features. The Source RS target is created using rejection sampling of the test data, see Section B.3.

B.3 Rejection sampling for creating shifted datasets

In experiments 5.2 and 5.3 we use rejection sampling for the oracle and source baselines, respectively. Let us briefly explain how this is achieved.

Let $\mathcal{D}$ be some dataset with distribution $p^0(X)$ that can be split into parts $p(X_{\bar{c}}|X_c)$ and $p(X_c)$. As noted in Section 3, we assume the latter changes (inducing a shift in distribution), while the former is fixed. We denote the shifted marginal distribution as $p^s(X_c)$ and the full distribution as $p^s(X) = p^s(X_c)p(X_{\bar{c}}|X_c)$.

In experiment 5.2, we desire a ground-truth target dataset for a given shift. We do not have data from $p^s(X)$, however we can use rejection sampling to *create* such dataset, which we denote by $\mathcal{D}^s$. Since we do not know $p^0(X)$ either, we sample from the empirical distribution, i.e. from data $\mathcal{D}$ itself, which will converge to the true distribution when $|\mathcal{D}|$ becomes large. To approximate $p^0(X_c)$, we train a simple KDE model and $p^s(X_c)$ is defined by shifting this distribution (see Section 4.2). This gives the following algorithm:

Algorithm 1: Rejection sampling from source dataset $\mathcal{D}$, given a predefined marginal shift T and desired test set size.

Given source dataset $\mathcal{D}$, shift T and desired shifted set size n_s
Fit density model $\hat{p}^0(X_c)$ to $\{\mathbf{x_c}|\mathbf{x} \in \mathcal{D}\}$
$\hat{p}^s(X_c) \leftarrow T(\hat{p}^0)(X_c)$
$M \leftarrow \max_{\mathbf{x_c} \in \mathcal{D}} \frac{\hat{p}^s(\mathbf{x_c})}{\hat{p}^0(\mathbf{x_c})}$
$\mathcal{D}^s \leftarrow \{\}$
while $|\mathcal{D}^s| < n_s$ **do**
  Sample $\mathbf{x}$ from $\mathcal{D}$ uniformly
  Sample $u \sim U(0, 1)$
  if $\frac{\hat{p}^s(\mathbf{x_c})}{\hat{p}^0(\mathbf{x_c})} > Mu$ **then**
    $\mathcal{D}^s \leftarrow \mathcal{D}^s \cup \{\mathbf{x}\}$
  end
end
return $\mathcal{D}^s$

In Experiment 5.2 we run the above with $\mathcal{D}$ an oracle test set. Since the oracle test set is very large, it covers p^0 relatively well. This allows us to approximate $p^0(X_c)$, and also means that draws from the empirical distribution are distributed approximately like the true underlying distribution.

In experiment 5.3 we use a similar set-up for creating baseline *Source (RS)* based on $\mathcal{D}_{test,f}$ alone, and to have a fair comparison we use rejection sampling to weigh the points.[7] In this case, however, the distribution $p^0(X_c)$, and in turn $p^s(X_c)$, cannot be approximated accurately. In addition, we may have very little data such that the same points need to be included many times (in regions with large $p^s(X_c)/p^0(X_c)$). As a result, although we see that *Source (RS)* performs better than unshifted *Source (all)*, it is a poor evaluation approach.

[7] Effectively, this reduces to an importance weighted estimate of the performance.

C GENERATIVE MODEL: REQUIREMENTS, TUNING AND RELIABILITY

This appendix focuses on the influence and choice of the generative model in the SYNG4ME framework. It consists of three parts. First, we explain how we choose and tune the generative model. Second, we include an ablation study that varies the size of the dataset SYNG4ME uses —showing when SYNG4ME does and does not improve model evaluation. Third, we extend on this and delve into reliability of SYNG4ME, by incorporating uncertainty into the SYNG4ME framework through using an ensemble of generative models.

C.1 CHOOSING AND TUNING THE GENERATIVE MODEL

Any generative model can be used to produce the synthetic test data, but some models may be better or worse than others. SYNG4ME uses CTGAN (Xu et al., 2019b), since this model is designed specifically for tabular data and has shown good performance. In this section, we explain how it is tuned, and include comparison to other generative models.

C.1.1 ASSESSING THE QUALITY OF GENERATIVE MODEL G IN SYNG4ME.

Approaches to model selection and quality assessment of generative models often measure the distance between the generated and the true distributions (Borji, 2019). In SYNG4ME, we use Maximum Mean Discrepancy (MMD) (Gretton et al., 2012), a popular choice for synthetic data quality (Sutherland et al., 2017; Bounliphone et al., 2016).

MMD performs a statistical test on distributions P^r (Real) and P^g (Generated), measuring the difference of their expected function values, with a lower MMD implying P^g is closer to P^r.

We use MMD in our auto-tuning and model selection step, comparing the generated data to a held-out test set, with G selected as the model with the lowest MMD. This step also serves to ensure that the data generated by SYNG4ME is indeed close to the real-world reference dataset of interest. Specifically, hyper-parameters of SYNG4ME when training G are tuned via a Tree-structured Parzen Estimator. We search over the number of epochs of training [100, 200, 300, 500], learning rate [2e-4, 2e-5, 2e-6] and embedding dimension [64,128,256]. For all methods, we have a small hyper-parameter validation set with size of 10% of the training dataset. Our objective is based on MMD minimization.

That said, of course alternative widely used metrics such as Inverse KL-Divergence or the Jensen-Shannon divergence could also be used as metrics of assessment.

C.1.2 INFLUENCE OF MODEL CHOICE.

Any generative model can be used as the core of SYNG4ME. For efficiency, a conditional generative model is highly desirable; this allows direct conditioning on subgroup or shift information, and not e.g. post-generation rejection sampling. Furthermore, some generative models may provide more or less realistic data. Here we compare SYNG4ME estimates provided by CTGAN, vs estimates given by TVAE and Normalizing Flows.

We assess these different base models for G on the race subgroup task from Sec. 5.1. Using SYNG4ME can assess the generative models based on MMD, but for completeness we also show inverse KL-divergence and Jensen-Shannon Divergence (JSD), where the metrics are computed vs a held-out validation dataset.

We show in Table 8, that the better quality metric does indeed translate into better performance when we use the synthetic data for model evaluation. We find specifically that CTGAN outperforms the other approaches, serving as validation for our selection.

Additionally, the results highlight that for practical application, one could evaluate the quality of G first using metrics such as MMD, inverse KLD or JSD, as a proxy for how well the generative model should perform.

We assume for the purposes of this experiment that the three classes of models are trained with the same optimization hyperparameters (epochs=200, learning rate=2e-4).

Table 8: Assessing the influence of model choice for G and illustrating how our quality assessment metrics in SYG4ME can be used to select the best model which indeed will provide the best performance. We see that indeed CTGAN performs best in this case.

Base G	MMD ↓	Inverse KLD ↑	JSD ↓	Subgroup (%)	Mean Absolute Error % ↓
CTGAN	**0.0014**	**0.995**	**0.03**	#1 (86%)	**0.28 ± 0.24**
				#2 (9%)	**17.64 ± 0.29**
				#3 (3%)	**2.96 ± 1.02**
				#4 (1%)	**1.14 ± 0.62**
				#5 (1%)	**1.03 ± 0.85**
NF	0.0034	0.970	0.09	#1 (86%)	16.25 ± 0.53
				#2 (9%)	26.35 ± 1.07
				#3 (3%)	20.50 ± 3.31
				#4 (1%)	27.14 ± 0.97
				#5 (1%)	26.04 ± 2.87
TVAE	0.4557	0.4987	0.505	#1 (86%)	25.93 ± 1.34
				#2 (9%)	35.40 ± 0.83
				#3 (3%)	24.05 ± 1.12
				#4 (1%)	33.0 ± 0.31
				#5 (1%)	37.69 ± 0.61

C.2 Understanding when to use SYNG4ME

Motivation. As discussed in Section 4, there is "no free lunch", since of course synthetic data cannot always help with model evaluation. We build on our experimental results to study the effect of test set sizes on SYNG4ME and in what conditions synthetic data is unnecessary or even harmful.

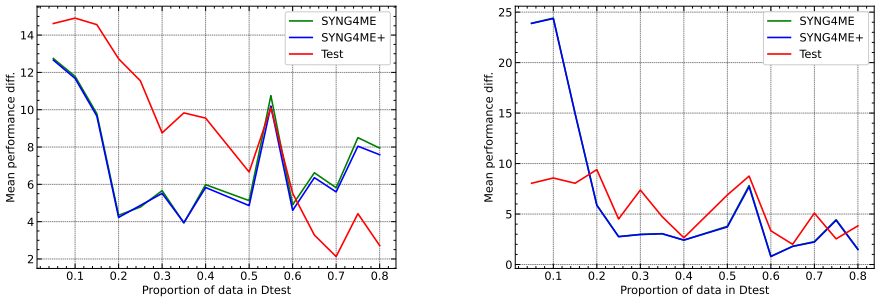
Setup. As is conventional, our datasets: $\mathcal{D}_{train,f}$ and $\mathcal{D}_{test,f}$ are split with proportion p from dataset $\mathcal{D}$. Recall, for validation we also have a large amount of unseen independent data $\mathcal{D}_{oracle}$, which can be thought of as representing the population seen by the model post-deployment. We then train the generative model G on the hold-out test dataset, i.e. $\mathcal{D}_{test,f}$.

In this experiment, we vary the proportion split p of $\mathcal{D}_{test,f}$ to study the effect of increasing the amount of test data for model evaluation. Since G is trained on this dataset, this also assesses the effect on the generator. Note that even in settings with a majority subgroup (many samples) and minority subgroup (fewer samples), that as the size of $\mathcal{D}_{test,f}$ increases, so will the number of samples contained in each subgroup.

Analysis. We quantify the effect of changing the proportion of $\mathcal{D}_{test,f}$ - i.e. as the proportion increases, the size of the test set correspondingly increases. We evaluate both the majority and minority-sized subgroups from the Adult dataset. The results in Fig. 7 provided as an example are for an RF-model. We interpret them as follows.

- Majority subgroup (many samples): When the proportion of data is lower (i.e. smaller test set), SYNG4ME is able to outperform test data alone. However, as the proportion gets large (i.e. more samples), we see that in fact simply having a large amount of real test data is better than synthetic data. That being said, at such high proportions we would have sufficient real-data for evaluation and hence typically have no need for synthetic data. Furthermore, since train-test splits typically have test proportions between 0.2-0.4, the large-portion situation is also unlikely in practice.
- Minority subgroup (fewer samples): This type of subgroup often does not have sufficient test data to adequately assess performance. This explains why it performs poorly for very low proportions $<= 0.1$ (very few samples) — there are insufficient data samples to train G. However, as the proportion increases (so does the number of samples), the performance improves drastically as we are able to better train G. This allows synthetic data to outperform real data alone, as we are able to generate more samples. Of course, for very large proportions (meaning we have a large number of samples), then synthetic data vs real test data are somewhat similar. Similar to the previous point, though such high test proportions are unlikely in practice.

Takeaway: Synthetic data using SYNG4ME should not be used when we have insufficient samples to train a generative model (MMD could assess this) or obtain good coverage. Alternatively, there



(a) Majority subgroup.

(b) Minority subgroup

Figure 7: Assessment of mean performance difference for SYNG4ME vs Test data for varying proportion data splits.

may be limited benefit of synthetic in high proportion settings — in any case, we typically have sufficient real test data in this case.

C.3 Uncertainty quantification to assess synthetic data

Evidently, on the basis of the previous experiment, we cannot always generate reliable synthetic data, which may lead to poor performance estimates. It is thus essential that we quantify the confidence of SYNG4ME estimates, so that we understand when to trust our predictions and when not. We quantify the uncertainty in the generative process using generative ensembles. We are motivated by the idea of Deep Ensembles (Lakshminarayanan et al., 2017), which has been shown to work well in the supervised domain (Rahaman et al., 2021; Lobacheva et al., 2020). We take a similar approach, but replace the predictive models by our generative models. First, we sample $\mathcal{D}^i_{train,G} \subset \mathcal{D}_{train,G}$, for $i = 1, ..., n_G$, with n_G the number of generators. Second, similar as before, we (i) train a generative model G^i on $\mathcal{D}^i_{train,G}$, (ii) generate $\mathcal{D}^i_{syn}$ and (iii) evaluate downstream models, repeating this for all i. The standard deviation in scores between different generators is used to quantify the uncertainty, with the average providing the final estimated score.

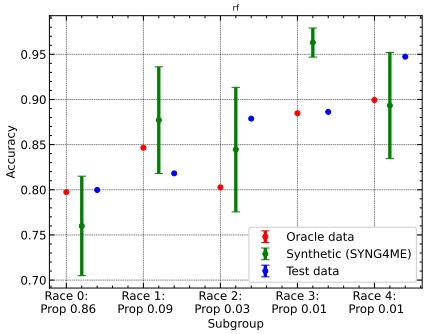
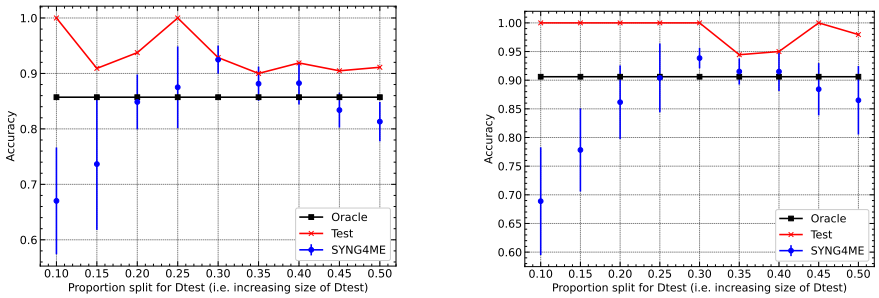


Figure 8: Example of uncertainty quantification using SYNG4ME.

Set-up. In order to train the deep generative ensemble, we randomly sample $\mathcal{D}^i_{train,G} \subset \mathcal{D}_{train,G}$, for $i = 1, ..., 5$. We then generate synthetic data for each subgroup using each generator and evaluate downstream model f performance, repeating for all i. The mean and standard deviation of scores across the ensemble represents the final estimate and uncertainty, respectively. Linked to the previous experiment, we assess how this translates to performance estimates for a large- and small-subgroup.

Analysis. We first show results for the subgroup setting considered in the main paper (Sec. 5.1). See Fig. 8 for the results. We observe that the SYNG4ME confidence bounds cover the true value well for most subgroups. Naturally, a traditional point estimate using just the test set does not provide such information. Note that for the majority group, the point estimate is decently accurate due to the



(a) Majority subgroup

(b) Minority subgroup

Figure 9: Analysis of how the uncertainty estimates of deep generative ensembles in SYGN4ME can be used to assess confidence of the SYNG4ME estimates

larger amount of samples available. This shows that SYNG4ME might not be necessary when plenty of test data is already available—e.g. for a majority group.

Next, we assess the setting with changes in proportions as in the previous experiment, i.e. Sec. C.2. We keep the range of proportions to 0.1-0.5, as this is the most realistic range in practice. We then assess the uncertainty of the ensemble of generators as a function of increasing proportion, with the RF downstream predictive model on Adult as an example. The results are illustrated in Fig. 9.

We see that for small proportions, the generative model produces intervals with large widths indicative of the uncertainty in the generative model, with the interval widths shrinking with increasing proportion. This result reflects the same pattern as the results from Sec. C.2.

In addition, the regions with small interval widths strongly reflect the regions for which synthetic data is beneficial (i.e. correct performance estimates), and vice versa, large intervals when it is not. As a consequence, end users could, in fact, use such intervals to help decide when to trust the performance estimates using SYNG4ME and when not.

Takeaway. SYNG4ME permits to quantify uncertainty of downstream predictive performance estimates using deep generative ensembles. The width of uncertainty intervals strongly reflect when the performance estimates from SYGN4ME are useful.

D ADDITIONAL EXPERIMENTS

This appendix includes a number of additional results. It consists of three parts. First, we illustrate how the definition of subgroup, allows two other interesting use cases for granular evaluation: (i) subgroups defined using a point of interest (i.e. how would a model perform on patients that look like X), and (ii) subgroups defined in terms of local density (i.e. how would a model perform on unlikely patients). Second, we include the results for SYNG4ME when we have access to the training set of predictor model f; since this dataset is usually larger, it can provide a higher quality model. Third, we include additional results for the main paper's experiments, using different downstream models, predictors and baselines.

D.1 OTHER DEFINITIONS OF SUBGROUPS

D.1.1 SUBGROUPS RELATING TO POINTS OF INTEREST

Motivation. End-users may also be interested in knowing how a model performs on some point of interest x^*. For example, a clinician may have access to a number of models and may need to decide for the specific patient in front of them what the model's potential predictive performance might be for the specific patient. This is relevant because global performance metrics may hide that models underperform for specific samples.

We can often assess model performance on samples similar to the sample of interest, i.e. "neighborhood performance". A challenge in reality is that we often only have access to a held-out test dataset (i.e. $\mathcal{D}_{test}$), rather than the entire population (i.e. $\mathcal{D}_{oracle}$). How then can we quantify performance?

Usually it is not possible to assess local performance using test data alone, since there may be very few samples that are similar enough—with "similar" defined in terms of some distance metric, i.e. $\mathcal{S} = \{x \in \mathcal{X} | d(x, x^*) < \epsilon\}$, with distance metric d and some small distance $\epsilon \geq 0$. As before, we can instead generate synthetic data in the region $\mathcal{S}$ and compute the performance on this set instead. This reduces the dependence on the small number of samples, in turn reducing variance in the estimated performance.

Set-up. We compare two approaches: (i) find nearest-neighbor points in $\mathcal{D}_{test}$)—which might suffer from limited similar samples—, or (ii) use SYNG4ME and generate synthetic samples $\mathcal{D}_{syn}$ in some neighborhood of x^*. Similarly as before, we assess these two methods by comparing estimates with a pseudo-ground truth that uses nearest neighbors on a much larger hold-out set, $\mathcal{D}_{oracle}$. Again, we compare (1) *mean absolute performance difference* and (2) *worst-case performance difference* between a specific evaluation set and the oracle dataset. We average across 10 randomly queried points x^* and use $k = 10$ nearest neighbours.

Analysis. The results in Table 9 shows that SYNG4ME has a much lower neighborhood performance gap, both average and worst case for x^* across models, when compared to the assessment using $\mathcal{D}_{test}$. The rationale is that by using synthetic data, we can generate more examples $\mathcal{D}_{syn}$ that closely resemble x^*, whereas with $\mathcal{D}_{test}$ we might be limited in the similar samples that can be queried - hence resulting in the higher variance estimates and poorer overall performance.

Table 9: Comparing two types of query methods to evaluate performance on points of interest x^*, which illustrates that SYNG4ME closer approximates an oracle both on average and worst case.

Model	Mean performance differenc ↓		Worst-case performance difference ↓	
	$\mathcal{D}_{syn}$ (SYNG4ME)	$\mathcal{D}_{test,f}$	$\mathcal{D}_{syn}$ (SYNG4ME)	$\mathcal{D}_{test,f}$
MLP	**0.083**	0.15	**0.482**	0.60
RF	**0.086**	0.18	**0.256**	0.50
GBDT	**0.093**	0.18	0.50	0.50

Takeaway. SYNG4ME's synthetic data can more robustly estimate performance on individual points of interest based on the samples generated in the neighborhood of x^*.

D.1.2 SUBGROUPS AS HIGH- AND LOW-DENSITY REGIONS

Motivation Models often perform worse on outliers and low-density regions due to the scarcity of data during training. We generate insight into this by defining subgroups in terms of density.

Methodology. We would like to partition the points into set of most likely to least likely w.r.t. density. We use the notion of α-support from (Alaa et al., 2022), namely:

$$\text{Supp}^{\alpha}(p) = \underset{S \subseteq \tilde{\mathcal{X}}}{\arg\min}\, V(S)\ s.t.\ p(\tilde{X} \in S) = \alpha, \tag{2}$$

with V some volume measure (e.g. Lebesgue). In other words, α-support $\text{Supp}^{\alpha}(p)$ denotes the smallest possible space to contain $\tilde{X}$ with probability α—which can be interpreted as the α most likely points. Subsequently, we can take a sequence of quantiles, $(q_i)_{i=0}^{k} \in [0, 1]$, with $q_i = \frac{i}{k}$ and look at the sequence of support sets, $(\text{Supp}^{q_i}(p))_{i=0}^{k}$.

The α-support itself always contains the regions with the highest density. To actually partition the points into sets from likely to unlikely, we instead look at the difference sets. That is, let us define sets $S_i = \text{Supp}^{q_i}(p) \backslash \text{Supp}^{q_{i-1}}(p)$ for $i = 1, ..., k$.

In fact, we do not know p exactly and even if we did, it is usually intractable to find an exact expression for the $\alpha-$support. Instead, we compute the α-support in the generative model's latent space, and not the original space. The latent distribution is usually chosen as a d_z-dimensional standard Gaussian, which has two advantages: (i) the distribution is continuous—cf. the original space, in which there may exist a lower-dimensional manifold on which all data falls; and (ii) the α-support is a simple sphere, with the radius given by $\text{CDF}^{-1}_{\chi^2_{d_z}}(\alpha)$.

Set-up. We train a generative model $G : \mathcal{Z} \to \tilde{\mathcal{X}}$ as before, where the input $\mathbf{z}$ is d_z-dimensional Gaussian noise. During generation we save inputs $\{z_j\}_{j=1}^{kN}$ and generate corresponding data $\mathcal{D}_{syn} = \{\tilde{x}_j\}_{j=1}^{kN}$. Sets for likely and unlikely points are defined using α-support in latent space $\mathcal{Z}$, for quantiles $\mathbf{q} = (q_i)_{i=1}^{k}$—see Section D.1.2. Specifically, let the index sets for each quantile set be given by $I_i = \{j | ||z_j||_2 < \text{Quantile}(\{||z_j||_2\}, q_i)\}$, giving synthetic sets $\mathcal{D}^{q_i}_{syn} = \{\tilde{x}_j \in \mathcal{D}_{syn} | j \in I_i\}$ for all i. We then evaluate the predictive model f on each $\mathcal{D}^{q_i}_{syn}$ and plot results w.r.t. $\mathbf{q}$. We use the SEER dataset.

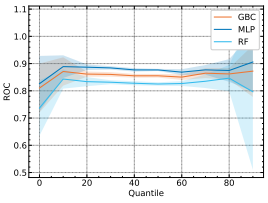


Figure 10: Performance on likely vs unlikely data, where error bars are over 10 runs

Analysis. Fig. 10 shows that for the tail quantiles (unlikely) data that the model performance is indeed highly variable and prone to possible poor performance. This is in contrast to the likely data, which has much more stable performance.

Takeaway. SYNG4ME's synthetic data can be used to quantify model performance on unlikely and likely data.

D.2 Improving G when training data of f is available

Motivation. In previous experiments, we have assumed that we have access to only $\mathcal{D}_{test,f}$ for training G. In some scenarios, we have access to $\mathcal{D}_{train,f}$. For example, the model developer has access to $\mathcal{D}_{train,f}$. Since the training dataset is often larger than the testing dataset, we could use this data to train G. Evidently, this results in some bias: we are now using synthetic data generated using G, which is trained on $\mathcal{D}_{train,f}$, to evaluate a predictive model that is also trained on $\mathcal{D}_{train,f}$. However, here we show that this bias is outweighed by the improved quality of G.

Setup. This experiment evaluates the mean performance difference for (i) SYNG4ME, (ii) SYNG4ME+ and (iii) $\mathcal{D}_{test,f}$. We follow the same setup as the granular subgroup experiment in Section 5.1, with the only difference being that $\mathcal{D}_{train,G} = \mathcal{D}_{train,f}$. We assess the utility of this setup for granular subgroup evaluation.

Analysis. Table 10 illustrates the performance similar to that of the main paper. We find that SYNG4ME mostly provides a more accurate evaluation of model performance (i.e. with estimates closer to the oracle) compared to a conventional hold-out dataset. This is especially true for the smaller subgroups for which synthetic data is indeed necessary. Furthermore, SYNG4ME has a lower worst-case error compared to $\mathcal{D}_{test,f}$. The results, with SYNG4ME+ also illustrate the benefit of augmenting real data with synthetic data, leading to lower performance differences.

Table 10: Mean absolute performance difference between predicted performance and performance evaluated by the oracle, where G is trained on $\mathcal{D}_{train,f}$. SYNG4ME better approximates true performance on minority subgroups, compared to test data alone.

Model	Subgroup (%)	Mean performance diff. ↓			Worst-case performance diff. ↓		
		SYNG4ME	SYNG4ME+	$\mathcal{D}_{test,f}$	SYNG4ME	SYNG4ME+	$\mathcal{D}_{test,f}$
RF	#1 (86%)	6.05 ± 0.72	2.33 ± 0.42	9.16 ± 4.48	7.08	2.84	11.65
	#2 (9%)	4.16 ± 0.33	4.29 ± 0.47	5.85 ± 3.18	4.44	4.82	8.94
	#3 (3%)	2.15 ± 1.04	1.88 ± 0.8	13.59 ± 5.34	3.41	3.03	19.73
	#4 (1%)	4.54 ± 0.33	4.54 ± 0.34	7.1 ± 3.93	4.89	4.96	13.23
	#5 (1%)	3.15 ± 0.77	3.06 ± 0.77	8.72 ± 0.0	4.2	4.12	8.72
GBDT	#1 (86%)	7.37 ± 1.09	3.25 ± 0.32	1.19 ± 0.81	9.22	3.6	2.42
	#2 (9%)	3.44 ± 0.52	3.37 ± 0.54	2.56 ± 1.08	3.98	4.02	4.49
	#3 (3%)	2.6 ± 1.53	2.46 ± 1.38	4.53 ± 1.04	4.29	3.97	5.73
	#4 (1%)	0.76 ± 0.58	0.76 ± 0.57	6.51 ± 5.07	1.31	1.34	14.53
	#5 (1%)	2.23 ± 0.82	2.15 ± 0.85	7.73 ± 3.33	3.47	3.45	9.4
MLP	#1 (86%)	6.72 ± 1.2	3.0 ± 0.46	1.01 ± 0.62	8.55	3.5	1.92
	#2 (9%)	4.32 ± 0.46	4.13 ± 0.54	2.45 ± 1.46	4.91	4.87	4.41
	#3 (3%)	1.98 ± 1.1	1.88 ± 0.99	4.06 ± 0.8	3.58	3.33	4.69
	#4 (1%)	2.57 ± 0.5	2.49 ± 0.53	7.44 ± 3.17	3.37	3.36	12.42
	#5 (1%)	2.17 ± 0.62	2.13 ± 0.64	4.3 ± 3.66	3.24	3.22	8.72
SVM	#1 (86%)	6.74 ± 0.58	3.0 ± 0.49	1.26 ± 0.93	7.37	3.6	2.46
	#2 (9%)	5.62 ± 0.14	5.31 ± 0.4	3.63 ± 1.25	5.83	5.81	5.63
	#3 (3%)	1.09 ± 0.73	1.04 ± 0.73	2.51 ± 1.09	2.43	2.33	3.88
	#4 (1%)	1.08 ± 0.4	0.97 ± 0.39	11.62 ± 4.31	1.51	1.44	16.01
	#5 (1%)	4.22 ± 0.6	4.17 ± 0.61	3.1 ± 2.99	5.2	5.18	6.71
AdaBoost	#1 (86%)	6.89 ± 0.96	3.06 ± 0.31	1.11 ± 0.71	8.61	3.44	2.14
	#2 (9%)	3.91 ± 0.3	3.76 ± 0.44	2.31 ± 1.86	4.24	4.31	5.15
	#3 (3%)	2.13 ± 1.43	2.01 ± 1.3	4.59 ± 2.48	4.22	4.0	7.79
	#4 (1%)	1.24 ± 0.65	1.16 ± 0.69	8.66 ± 6.46	1.99	1.97	19.89
	#5 (1%)	3.42 ± 0.71	3.36 ± 0.73	5.17 ± 3.55	4.47	4.44	8.05
Bagging	#1 (86%)	5.81 ± 1.02	2.65 ± 0.53	10.01 ± 4.49	7.03	3.61	12.83
	#2 (9%)	4.89 ± 0.68	5.06 ± 0.44	7.65 ± 3.22	5.97	5.54	10.69
	#3 (3%)	3.72 ± 1.65	3.37 ± 1.64	8.5 ± 4.77	5.98	5.67	14.55
	#4 (1%)	7.01 ± 0.28	7.0 ± 0.3	6.16 ± 3.51	7.46	7.48	11.21
	#5 (1%)	5.06 ± 1.09	5.0 ± 1.09	4.41 ± 4.07	6.59	6.55	9.4
LR	#1 (86%)	6.51 ± 0.63	2.97 ± 0.3	1.09 ± 0.41	7.57	3.38	1.81
	#2 (9%)	4.11 ± 0.32	3.9 ± 0.51	3.33 ± 0.98	4.52	4.52	4.57
	#3 (3%)	1.15 ± 0.73	1.11 ± 0.64	4.6 ± 2.72	2.28	2.08	7.61
	#4 (1%)	1.69 ± 0.59	1.62 ± 0.62	7.16 ± 4.27	2.57	2.55	13.48
	#5 (1%)	4.08 ± 0.76	4.02 ± 0.78	4.76 ± 3.21	5.44	5.41	7.38

D.3 Additions to main paper results

D.3.1 Subgroup performance evaluation w/ more downstream models

Motivation. The performance of the model, on subgroups, is likely influenced by the class of downstream predictive model f. We aim to assess the performance of the granular subgroups for a broader class of downstream models f

Setup. This experiment evaluates the mean performance difference for (i) SYNG4ME, (ii) SYNG4ME+ and (iii) $\mathcal{D}_{test,f}$. We follow the same setup as the granular subgroup experiment in Section 5.1. We increase the predictive models beyond RF, MLP and GBDT and further include: SVM, AdaBoost, Bagging Classifier, and Logistic Regression.

Analysis. Table 11 illustrates that SYNG4ME, when evaluated with more models, still better approximates the true performance on minority subgroups, compared to test data alone. This is in terms of Mean absolute performance difference between predicted performance and performance evaluated by the oracle.

D.3.2 Subgroup worst-case performance evaluation

Motivation. When estimating sub-group performance, of course we want to have as low error as possible on average (i.e. low mean performance difference). That said, average performance glosses over the worst-case scenario. We desire that the worst-case mean performance difference is also low. This is to ensure that, by chance, the performance estimates are not wildly inaccurate. This scenario is particularly relevant, as by chance the testing data could either over- or under-estimate model performance, leading us to draw incorrect conclusions.

Table 11: Mean absolute performance diff. between the predicted performance and the oracle for a broader class of models

Model	Subgroup (%)	Mean absolute performance diff. % ↓		
	Race	SYNG4ME	SYNG4ME+	$\mathcal{D}_{test,f}$
RF	#1 (86%)	7.26 ± 0.94	2.31 ± 1.56	10.02 ± 3.36
	#2 (9%)	4.33 ± 0.34	4.55 ± 0.38	6.83 ± 2.67
	#3 (3%)	3.48 ± 0.82	2.98 ± 0.79	13.68 ± 4.39
	#4 (1%)	1.14 ± 0.62	1.18 ± 0.62	7.26 ± 3.79
	#5 (1%)	1.03 ± 0.85	0.96 ± 0.84	8.06 ± 2.0
GBDT	#1 (86%)	7.47 ± 0.83	2.97 ± 0.85	1.39 ± 0.98
	#2 (9%)	4.25 ± 0.36	4.07 ± 0.31	2.14 ± 1.0
	#3 (3%)	4.40 ± 0.91	4.16 ± 0.91	4.39 ± 1.92
	#4 (1%)	1.61 ± 0.62	1.61 ± 0.65	4.50 ± 4.73
	#5 (1%)	0.68 ± 0.58	0.68 ± 0.56	6.31 ± 3.29
MLP	#1 (86%)	6.79 ± 1.09	2.85 ± 0.74	1.07 ± 0.83
	#2 (9%)	5.06 ± 0.37	4.72 ± 0.27	1.63 ± 1.35
	#3 (3%)	3.60 ± 0.82	3.43 ± 0.83	4.75 ± 0.94
	#4 (1%)	0.55 ± 0.29	0.57 ± 0.31	6.21 ± 3.44
	#5 (1%)	0.48 ± 0.53	0.47 ± 0.54	4.46 ± 2.85
SVM	#1 (86%)	6.80 ± 0.94	2.90 ± 0.89	1.22 ± 0.87
	#2 (9%)	4.66 ± 0.63	4.37 ± 0.48	2.57 ± 1.63
	#3 (3%)	2.15 ± 0.94	2.05 ± 0.97	3.59 ± 1.91
	#4 (1%)	2.86 ± 0.79	2.89 ± 0.82	7.58 ± 5.68
	#5 (1%)	3.19 ± 0.87	3.17 ± 0.87	3.63 ± 2.93
AdaBoost	#1 (86%)	6.75 ± 1.2	2.8 ± 0.81	1.12 ± 0.95
	#2 (9%)	4.66 ± 0.38	4.36 ± 0.28	1.52 ± 1.57
	#3 (3%)	3.96 ± 0.95	3.74 ± 0.98	4.95 ± 2.21
	#4 (1%)	1.15 ± 0.71	1.19 ± 0.75	6.77 ± 5.34
	#5 (1%)	1.57 ± 0.77	1.55 ± 0.77	4.40 ± 3.08
Bagging	#1 (86%)	7.30 ± 1.23	2.77 ± 1.65	10.97 ± 3.40
	#2 (9%)	3.58 ± 0.44	3.97 ± 0.58	8.50 ± 2.66
	#3 (3%)	4.28 ± 0.62	3.90 ± 0.60	8.66 ± 4.43
	#4 (1%)	1.85 ± 0.81	1.92 ± 0.77	9.33 ± 5.07
	#5 (1%)	4.48 ± 0.8	4.41 ± 0.78	5.86 ± 3.70
LR	#1 (86%)	6.62 ± 0.98	2.76 ± 0.86	1.13 ± 0.76
	#2 (9%)	4.62 ± 0.35	4.30 ± 0.28	2.00 ± 1.54
	#3 (3%)	2.68 ± 0.92	2.52 ± 0.93	5.49 ± 2.40
	#4 (1%)	0.89 ± 0.45	0.93 ± 0.45	5.18 ± 4.43
	#5 (1%)	2.29 ± 0.77	2.27 ± 0.76	4.32 ± 2.98

Setup. This experiment evaluates the worst-case mean performance difference for (i) SYNG4ME, (ii) SYNG4ME+ and (iii) $\mathcal{D}_{test,f}$. We follow the same setup as the granular subgroup experiment in Section 5.1.

Analysis. Table 12 illustrates that SYNG4ME and the augmented SYGM4ME+ have a lower worst-case performance compared to evaluation with real test data. This further shows that, by chance, evaluation with real data can severely over- or under-estimate performance, leading to incorrect conclusions about the model's abilities. SYNG4ME's lower worst-case error, means even in the worst scenario, that SYNG4ME's estimates are still closer to true performance.

D.3.3 Intersectional performance matrix deep-dive

Motivation. We perform a deep-dive of the intersectional performance matrices generated by SYNG4M3E, simply using test data alone and the oracle.

Analysis. We first see in Table 13 that, in general, the intersectional matrix SYGN4ME has a much *lower* error when estimating performance for intersectional subgroups. i.e. this is more similar to the oracle, compared to using $\mathcal{D}_{test,f}$. Note that we set the minimum number of samples required for

Table 12: Worst-case performance difference between the predicted performance and performance evaluated by the oracle. The worst case is over 10 runs

Model	Subgroup (%)	Worst-case performance diff. % ↓		
	Race	SYNG4ME	SYNG4ME+	$\mathcal{D}_{test,f}$
RF	#1 (86%)	10.473	4.181	11.65
	#2 (9%)	4.710	5.061	8.936
	#3 (3%)	3.136	2.652	19.732
	#4 (1%)	2.167	2.205	13.228
	#5 (1%)	1.641	1.579	8.725
GBDT	#1 (86%)	10.065	5.000	2.422
	#2 (9%)	4.602	4.592	4.487
	#3 (3%)	4.815	4.487	5.727
	#4 (1%)	2.853	2.858	14.535
	#5 (1%)	1.470	1.455	9.396
MLP	#1 (86%)	9.791	4.711	1.920
	#2 (9%)	5.130	5.070	4.410
	#3 (3%)	4.340	4.060	4.694
	#4 (1%)	0.987	1.04	12.418
	#5 (1%)	1.290	1.280	8.725
SVM	#1 (86%)	10.107	5.019	2.462
	#2 (9%)	5.436	5.369	5.632
	#3 (3%)	3.012	2.823	3.883
	#4 (1%)	4.057	4.153	16.005
	#5 (1%)	3.103	3.094	6.711
AdaBoost	#1 (86%)	9.953	5.234	2.144
	#2 (9%)	4.859	4.883	5.149
	#3 (3%)	4.273	3.909	7.789
	#4 (1%)	1.811	1.965	19.888
	#5 (1%)	2.712	2.698	8.054
Bagging	#1 (86%)	10.070	4.666	12.827
	#2 (9%)	4.189	4.332	10.694
	#3 (3%)	4.657	4.230	14.554
	#4 (1%)	3.282	3.246	11.21 0
	#5 (1%)	5.476	5.437	9.396
LR	#1 (86%)	8.719	4.704	1.811
	#2 (9%)	5.189	5.137	4.569
	#3 (3%)	2.691	2.539	7.611
	#4 (1%)	1.534	1.635	13.477
	#5 (1%)	2.833	2.822	7.383

validation = 100 samples. This induces sparseness, of course, but is necessary in order to prevent evaluation on too few data points. However, in cases where $\mathcal{D}_{test,f}$ does not have data for the intersection (i.e. $n < 100$), we do not consider these NaN blocks as part of our calculation; in fact, this makes it easier for $\mathcal{D}_{test,f}$.

The rationale is evident when evaluating the intersectional performance matrices for each group. We present the following findings.

- **SYNG4ME's insights are correct**: the underperforming subgroups, as noted by SYNG4ME, match the oracle. Therefore, it serves as a further validation.

- $\mathcal{D}_{test,f}$ **is very sparse after cut-offs**: the 100 sample cut-off highlights the key challenge of evaluation on a test set. We may not have sufficient samples for each intersection to perform an evaluation.

Table 13: Adult: Intersectional performance matrix difference vs the oracle

Model	SYNG4ME	$\mathcal{D}_{test,f}$
Average	0.13 ± 0.005	0.21 ± 0.002
RF	0.133	0.211
GBDT	0.128	0.207
MLP	0.128	0.207
SVN	0.138	0.209
AdaBoost	0.126	0.206
Bagging	0.138	0.211
LR	0.126	0.207

D.3.4 COVID-19 DATA: SUBGROUP & INTERSECTIONAL PERFORMANCE EVALUATION

Motivation. In the main paper, we have performed an evaluation of (P1) Reliable granular evaluation using the Adult dataset. We extend this to the Covid-19 dataset (Baqui et al., 2020) and assess the ethnicity subgroup of Brazilian patients that has known variation, due to their representational differences.

Setup. We use the same set-up and evaluation metrics as in Section 5.1.

Analysis. Table 14 shows the results for the Covid-19 dataset. We find that SYNG4ME mostly provides a more accurate evaluation of model performance (i.e. with estimates closer to the oracle) compared to a conventional hold-out dataset. In addition, we find that SYNG4ME often has lower worst-case performance. However, we note that given the small size of the data set, we have fewer than 1000 samples to train G. We hypothesize that this result could be improved if we had more data.

We additionally find that for the *intersectional performance matrix* that SYNG4ME improves estimates, with mean absolute error of $\mathbf{0.16 \pm 0.043}$ lower than for $\mathcal{D}_{test,f}$ of $\mathbf{0.22 \pm 0.06}$. We present the intersectional performance matrix for the Covid-19 dataset in Fig. 11.

Table 14: Covid-19 results: SYNG4ME better approximates true performance on minority subgroups.

Model	Subgroup (%)	Mean performance diff. ↓			Worst-case performance diff. ↓		
		SYNG4ME	SYNG4ME+	$\mathcal{D}_{test,f}$	SYNG4ME	SYNG4ME+	$\mathcal{D}_{test,f}$
RF	#1 (59%)	7.859	4.752	16.693	8.64	5.186	20.494
	#2 (32%)	15.697	12.974	16.641	17.308	14.4	17.949
	#3 (6%)	12.584	12.102	15.733	14.084	13.484	22.147
	#4 (2%)	11.526	11.294	29.977	12.509	12.246	33.333
	#5 (<0%)	18.477	18.477	70.0	20.525	20.525	70.0
GBDT	#1 (59%)	6.772	5.663	1.996	7.397	6.151	4.732
	#2 (32%)	15.635	13.887	5.089	17.794	15.692	7.703
	#3 (6%)	12.696	12.53	6.792	15.582	15.553	13.929
	#4 (2%)	15.407	15.28	14.021	17.347	17.225	20.671
	#5 (<0%)	16.365	16.365	70.0	17.323	17.323	70.0
MLP	#1 (59%)	6.403	5.686	2.133	6.937	6.573	6.524
	#2 (32%)	14.667	13.115	3.788	16.726	14.975	4.579
	#3 (6%)	9.651	9.591	7.912	11.203	11.048	16.959
	#4 (2%)	16.173	16.036	11.099	18.459	18.328	21.861
	#5 (<0%)	22.687	22.687	70.0	24.537	24.537	70.0
SVM	#1 (59%)	4.78	4.081	2.284	5.899	5.013	6.09
	#2 (32%)	7.391	6.507	3.108	9.271	8.146	7.778
	#3 (6%)	6.914	6.908	6.324	9.099	9.15	11.895
	#4 (2%)	9.865	9.753	10.688	10.872	10.756	20.238
	#5 (<0%)	4.31	4.31	60.0	5.742	5.742	60.0
AdaBoost	#1 (59%)	7.016	5.894	1.866	7.551	6.309	4.393
	#2 (32%)	16.447	14.873	2.214	17.584	16.13	3.742
	#3 (6%)	13.017	12.984	12.226	14.187	14.176	19.651
	#4 (2%)	17.617	17.492	7.463	18.914	18.782	16.667
	#5 (<0%)	10.089	10.089	50.0	10.771	10.771	50.0
Bagging	#1 (59%)	7.777	4.605	17.284	8.978	5.59	20.287
	#2 (32%)	13.531	11.022	16.257	14.658	12.119	17.778
	#3 (6%)	12.118	11.629	16.872	12.913	12.44	24.859
	#4 (2%)	14.752	14.558	20.603	15.779	15.558	30.952
	#5 (<0%)	3.686	3.686	60.0	5.072	5.072	60.0
LR	#1 (59%)	6.044	5.235	1.444	6.667	5.856	3.969
	#2 (32%)	14.833	13.425	1.892	16.348	14.808	2.768
	#3 (6%)	7.992	7.926	9.138	9.298	9.217	14.859
	#4 (2%)	12.046	11.923	11.424	13.495	13.37	20.238
	#5 (< 0%)	25.743	25.743	70.0	27.689	27.689	70.0

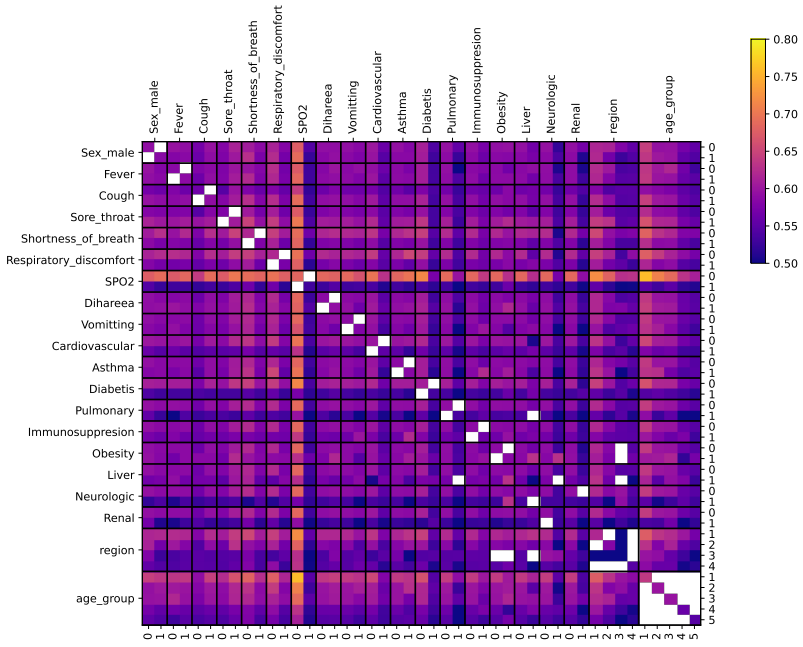


Figure 11: Covid-19 intersectional performance matrix

D.3.5 SENSITIVITY TO SHIFTS ON ADDITIONAL FEATURES

Motivation. In the main paper, we have characterized the sensitivity across operating ranges for two features in two datasets (Adult and SEER). Ideally, a practitioner would like to understand the sensitivity to all features in the dataset. We now conduct this assessment on the Adult dataset, producing model sensitivity curves for all features.

Setup. The setup is the same as in Section 5.2.

Analysis. We include the model sensitivity curves for all features as part of the example model report in Appendix E, see Figures 14, (a)-(t).

D.4 FAIRNESS METRICS: SUBGROUP EVALUATION

Motivation. We have primarily studied reliable estimation of model performance on different subgroups. This is easily generalized to estimate fairness metrics of ML models on specific subgroups. This can provide further insight into the use of synthetic data for model testing.

Setup. This experiment evaluates the mean performance difference for (i) SYNG4ME, (ii) SYNG4ME+ and (iii) $\mathcal{D}_{test,f}$. We follow the same setup as the granular subgroup experiment in Section 5.1. We evaluate a RF model. We assess the following fairness metrics: (i) Disparate Impact (DI) ratio (demographic parity ratio) and (ii) Equalized-Odds (EO) ratio. When estimating these metrics for each subgroup (e.g. race group), we then condition on sex as the sensitive attribute.

The DI ratio is: ratio between the smallest and the largest group-level selection rate $E[f(X)|A = a]$, across all values of the sensitive feature(s) $a \in A$.

The EO ratio is the smaller of two metrics between TPR ratio (smallest and largest of $P[f(X) = 1|A = a, Y = 1]$, across all values of the sensitive feature(s)) and FPR ratio (similar but defined for $P[f(X) = 1|A = a, Y = 0]$), , across all values of the sensitive feature(s) $a \in A$.

Analysis. Table 15 illustrates that SYNG4ME's performance on both fairness metrics, better approximates the true oracle metric on minority subgroups, compared to test data alone. This is in terms of Mean absolute performance difference between predicted performance and performance evaluated by the oracle.

We also assess the worst-case scenario as well, as done previously.

Table 15: Mean absolute diff. between the predicted metric and the oracle metric. Fairness metrics: Disparate Impact (DI) and Equalized-Odds (EO). Averaged over 5 runs.

Metric	Subgroup (%)	Mean absolute performance diff. % ↓		
	Race	SYNG4ME	SYNG4ME+	$\mathcal{D}_{test,f}$
DI	#1 (86%)	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01
	#2 (9%)	0.03 ± 0.01	0.02 ± 0.01	0.02 ± 0.02
	#3 (3%)	0.10 ± 0.02	0.10 ± 0.02	0.12 ± 0.09
	#4 (1%)	0.09 ± 0.01	0.09 ± 0.01	0.12 ± 0.06
	#5 (1%)	0.03 ± 0.01	0.03 ± 0.01	0.05 ± 0.04
EO	#1 (86%)	0.18 ± 0.08	0.02 ± 0.02	0.09 ± 0.07
	#2 (9%)	0.09 ± 0.09	0.08 ± 0.04	0.53 ± 0.22
	#3 (3%)	0.29 ± 0.08	0.31 ± 0.08	0.47 ± 0.04
	#4 (1%)	0.12 ± 0.08	0.12 ± 0.06	0.28 ± 0.24
	#5 (1%)	0.38 ± 0.02	0.38 ± 0.02	0.46 ± 0.0

Table 16 illustrates that SYNG4ME and the augmented SYGM4ME+ have a lower worst-case estimated difference compared to evaluation with real test data. This further shows that, by chance, evaluation with real data can over- or under-estimate fairness, leading to incorrect conclusions about the model's abilities. SYNG4ME's lower worst-case error, means even in the worst scenario, that SYNG4ME's estimates are still closer to true fairness metric.

Table 16: Worst-case performance difference between the predicted metric and metric evaluated by the oracle. Fairness metrics: Disparate Impact (DI) and Equalized-Odds (EO). The worst case is over 5 runs

Metric	Subgroup (%)	Worst-case performance diff. % ↓		
	Race	SYNG4ME	SYNG4ME+	$\mathcal{D}_{test,f}$
DI	#1 (86%)	0.033	0.02	0.021
	#2 (9%)	0.036	0.034	0.070
	#3 (3%)	0.118	0.124	0.216
	#4 (1%)	0.102	0.102	0.193
	#5 (1%)	0.035	0.036	0.099
EO	#1 (86%)	0.305	0.056	0.221
	#2 (9%)	0.240	0.147	0.851
	#3 (3%)	0.377	0.393	0.549
	#4 (1%)	0.245	0.237	0.750
	#5 (1%)	0.400	0.401	0.462

D.5 INCORPORATING PRIOR KNOWLEDGE ON SHIFT: COVID-19

Motivation. We have the ability of assessing distributional shift where we have *some* knowledge of the shifted distribution. Specifically, here we assume we only observe a few of the features, from the target domain.

Setup. Our setup in similar to Section 5.2.2, however on a different dataset - i.e. Covid-19. There are known distributional differences between the north and south of Brazil. For example: more different prevalence of respiratory issues, sex proportions, obesity rates etc. Hence we train the predictive model on patients from the South (larger population) and seek to evaluate potential performance on patients from the North. We take the largest sub-regions for each.

To validate our estimate, we use the actual northern dataset (Target) as ground-truth. Our baselines are as in Section 5.2.2. Since the features are primarily binary, we parameterise the distributions as binomial with a probability of prevalence for their features. We can then sample from this distribution.

Analysis. Fig. 12 shows the average estimated performance of f, as a function of the number of features observed from the target dataset. We see that the SYNG4ME estimates are closer to the oracle across the board compared to baselines. Furthermore, for increasing number of features (i.e. increasing prior knowledge), we observe that SYNG4ME estimates converge to the oracle.

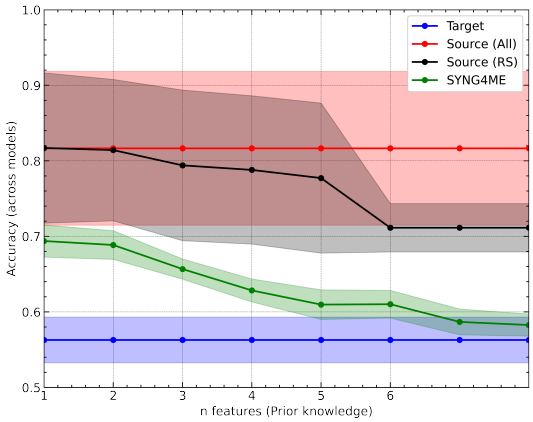


Figure 12: $\mathcal{D}_{syn}$ is better able to approximate performance in the target domain compared to baselines and that performance improves as more prior knowledge is incorporated via added features. Points are connected to highlight trends.

E EXAMPLE MODEL REPORT

Below we present an example of the type of model report that could be produced when evaluating models using using SYNG4ME.

Dataset: Adult (Asuncion & Newman, 2007).

Intersectional model performance matrix: diagnosing at a granular level.

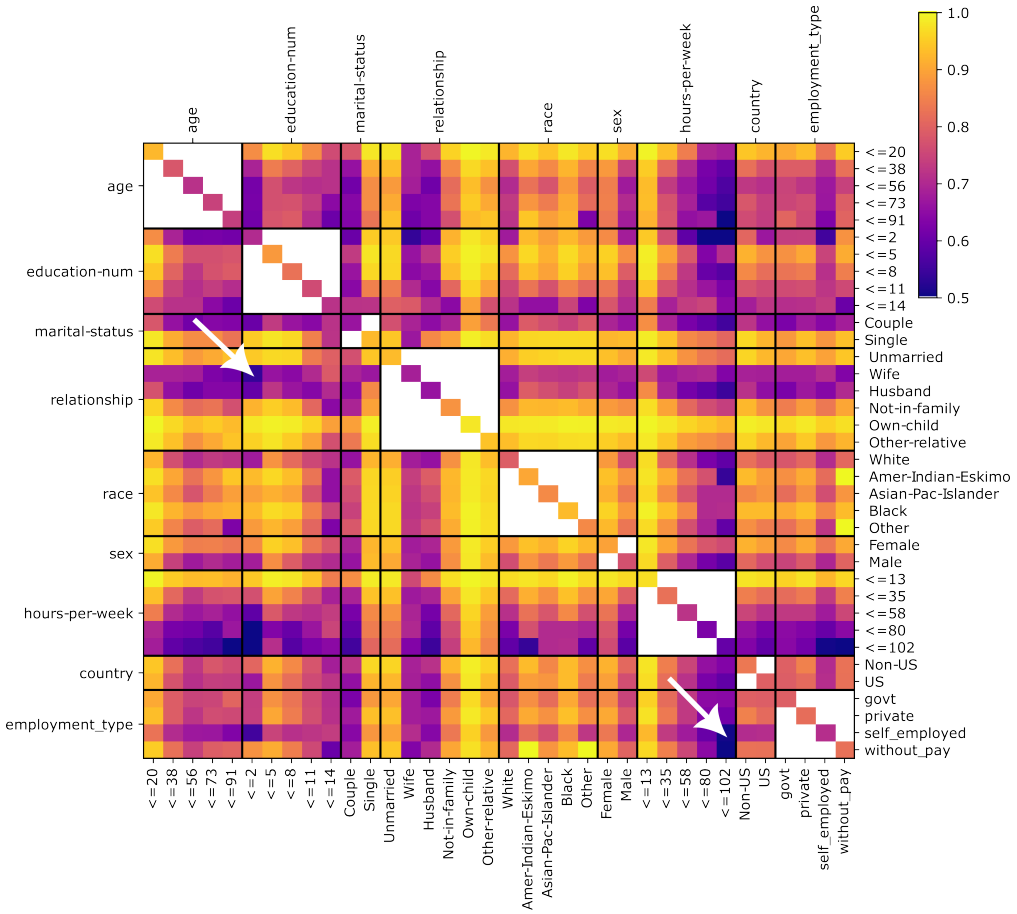
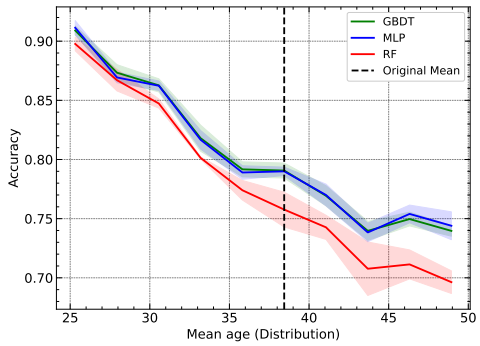


Figure 13: Intersectional performance matrix for the RF model, which diagnoses underperforming 2-feature subgroups (darker implies underperformance).

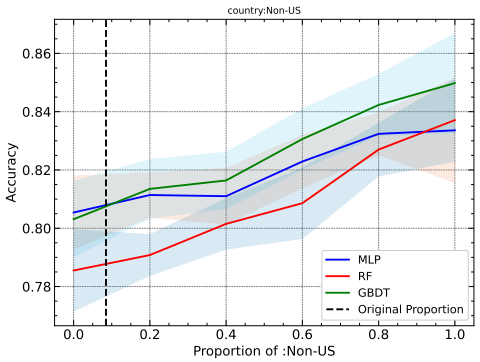
INSIGHT 1: Model underperformance on married females with 2 or less years of education. (top left arrow)

INSIGHT 2: Model underperformance on self-employed who work more than 80 hours a week. (bottom right arrow)

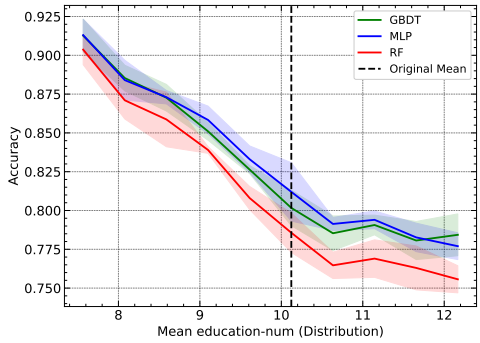
Model sensitivity curves: helping to understand performance trends for shifts across the operating range:



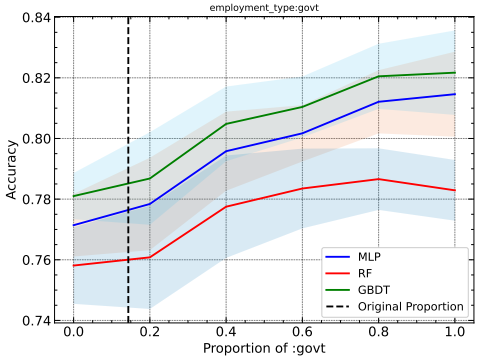
(a) Age: Performance decreases as mean age increases.



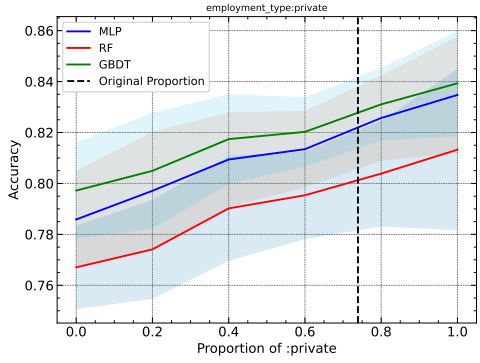
(b) Country of origin: performance increases as the proportion of non-US individuals increases



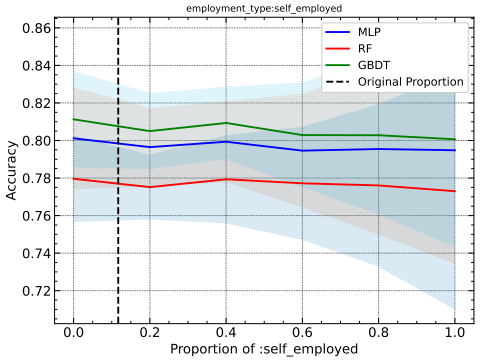
(c) Number of years of education: Performance decreases as mean number of education years increases.



(d) Employment type (government): performance increases as the proportion of government employed individuals increases

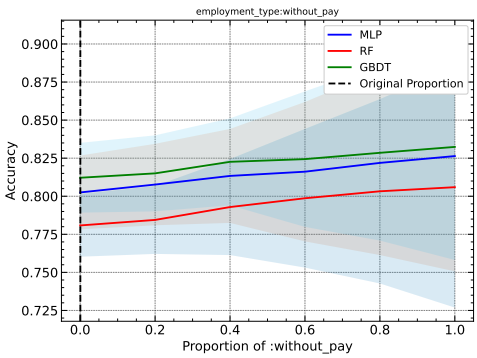


(e) Employment type (private): performance increases as the proportion of private employed individuals increases

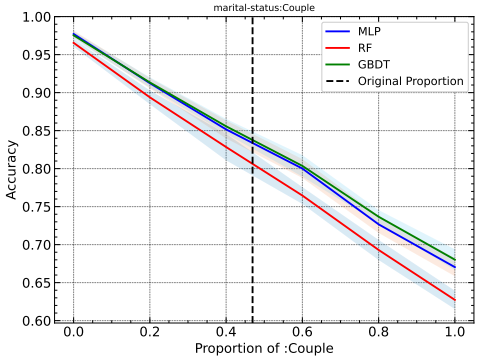


(f) Employment type (self-employed): performance is consistent even as proportion of self-employed individuals increases

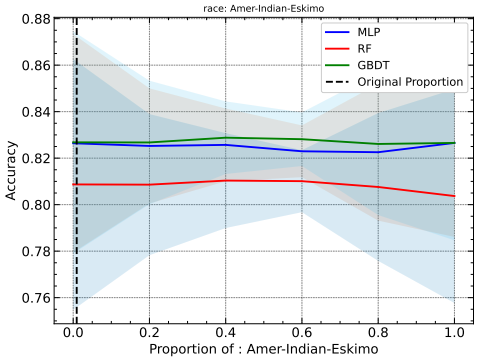
Figure 14: Model sensitivity curves for different features, illustrating the relationships/model performance across the operating range.



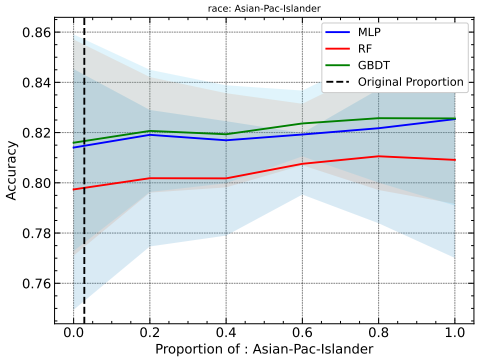
(g) Employment type (without pay): performance is consistent even as proportion of without-pay individuals increases



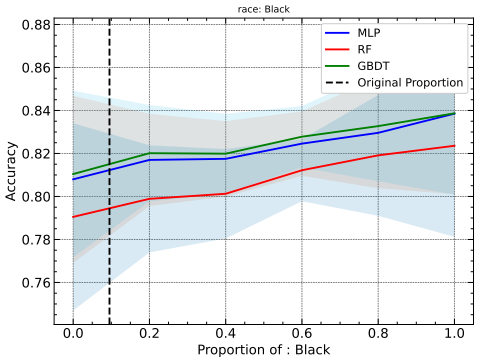
(h) Marital status (couple): performance decreases as the proportion of married individuals increases



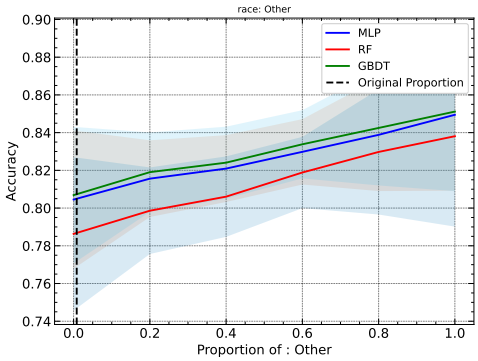
(i) Race (American-Indian-Eskimo): performance remains consistent even as the proportion of American-Indian-Eskimo individuals increases



(j) Race (Asian-Pacific-Islander): performance remains consistent even as the proportion of Asian-Pacific-Islander individuals increases

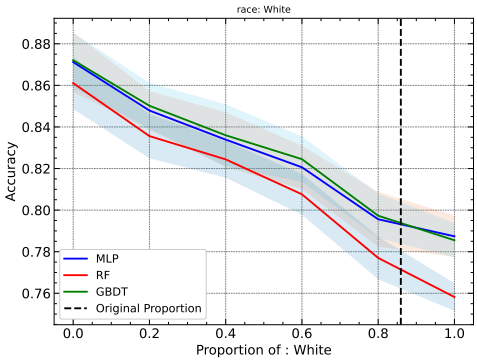


(k) Race (Black): performance increases as the proportion of black individuals increases

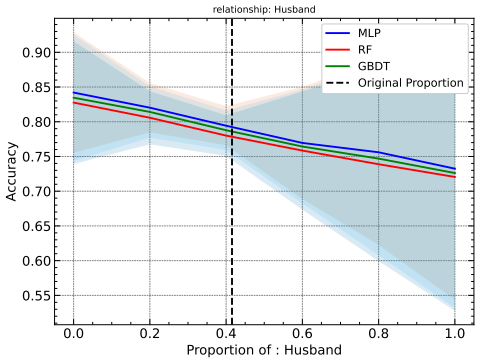


(l) Race (Other): performance increases as the proportion of other individuals increases

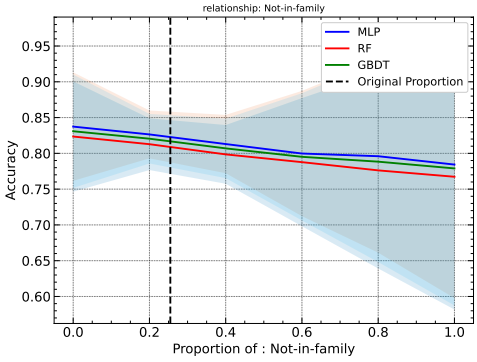
Figure 14: Model sensitivity curves for different features, illustrating the relationships/model performance across the operating range.



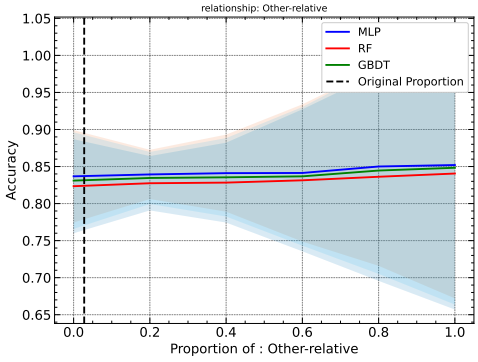
(m) Race (White): performance decreases as the proportion of white individuals increases



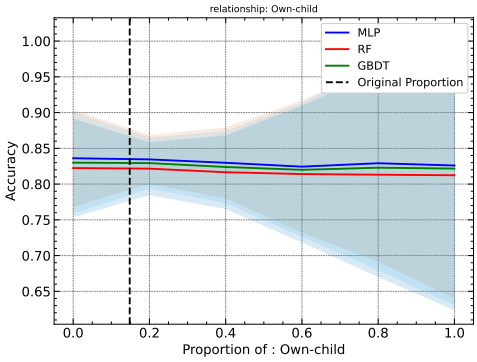
(n) Relationship (Husband): performance decreases as the proportion of individuals classed as husbands increases



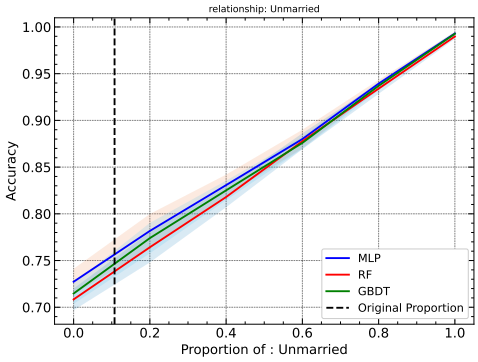
(o) Relationship (Not-in-family): performance decreases as the proportion of individuals classed as Not-in-family increases



(p) Relationship (other-relative): performance remains consistent as the proportion of individuals classed as other-relative increases

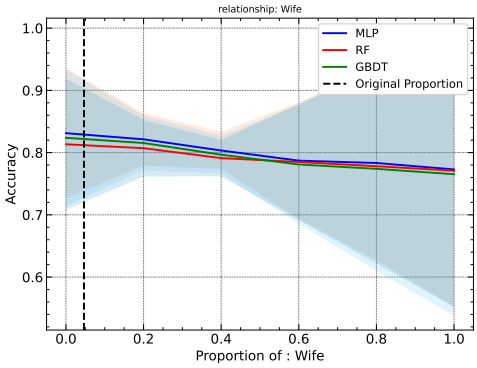


(q) Relationship (Own-child): performance remains consistent as the proportion of individuals classed as Own-child increases

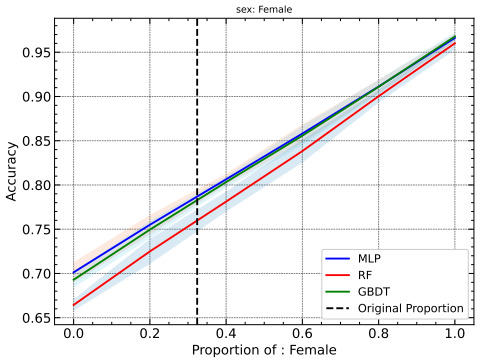


(r) Relationship (Unmarried): performance increases as the proportion of individuals classed as Unmarried increases

Figure 14: Model sensitivity curves for different features, illustrating the relationships/model performance across the operating range.



(s) Relationship (Wife): performance decreases as the proportion of individuals classed as Wife increases



(t) Sex (Female): performance increases as the proportion of females increases

Figure 14: Model sensitivity curves for different features, illustrating the relationships/model performance across the operating range.