Data Frequency Coverage Impact on AI Performance

Erin Lanus*, Brian Lee*, Jaganmohan Chandrasekaran†, Laura J. Freeman*,
M S Raunak ‡, Raghu N. Kacker ‡, D. Richard Kuhn ‡,
* National Security Institute, Virginia Tech, Arlington, VA, USA
† Sanghani Center for Artificial Intelligence & Data Analytics, Virginia Tech, Arlington, VA, USA
‡ Information Technology Lab, National Institute of Standards and Technology, MD, USA

***Abstract*—Artificial Intelligence (AI) models use statistical learning over data to solve complex problems for which straightforward rules or algorithms may be difficult or impossible to design; however, a side effect is that models that are complex enough to sufficiently represent the function may be uninterpretable. Combinatorial testing, a black-box approach arising from software testing, has been applied to test AI models. A key differentiator between traditional software and AI is that many traditional software faults are deterministic, requiring a failure-inducing combination of inputs to appear only once in the test set for it to be discovered. On the other hand, AI models learn statistically by reinforcing weights through repeated appearances in the training dataset, and the frequency of input combinations plays a significant role in influencing the model's behavior. Thus, a single occurrence of a combination of feature values may not be sufficient to influence the model's behavior. Consequently, measures like simple combinatorial coverage that are applicable to software testing do not capture the frequency with which interactions are covered in the AI model's input space. In this work, we develop methods to characterize the data frequency coverage of feature interactions in training datasets and analyze the impact of imbalance, or skew, in the combinatorial frequency coverage of the training data on model performance. We demonstrate our methods with experiments on the CDC Diabetes Health Indicators open-source dataset using several classical machine learning algorithms.**

***Index Terms*—Testing AI; Combinatorial Coverage; Combinatorial Frequency.**

I. Introduction

Combinatorial testing and combinatorial coverage help design and measure the completeness of test suites in software testing. Unlike most traditional software testing adequacy criteria that depend on structural coverage of the source code, combinatorial testing focuses on ensuring that combinations of input parameters and their values are utilized to adequately test the system. Studies have shown that most software bugs result from the interaction of only a few parameters and their specific values [1]. Thus systematically covering all t-way (where $t = 2, 3, 4$ etc.) parameter value combinations results in an efficient and effective testing of the system. The core idea of combinatorial coverage is to measure how well a test suite covers all valid t-way combinations of parameters or "t-tuples" (e.g., pairwise/2-way or 3-way combinations) for a given strength of interaction t.

The direct application of combinatorial testing as it is used in software verification and validation may not work while testing artificial intelligence or machine learning (AI/ML) based systems. For example, the use of covering arrays for combinatorial testing in software testing relies on the assumption that the software faults are deterministic and so a fault-inducing interaction of parameters will always trigger a fault, if the system is in a particular state. Covering arrays by definition are required to cover each t-way interaction at least once but redundant coverage is typically discouraged as each row in the array corresponds to a test consuming resources to execute. The statistical learning algorithms through which AI models are trained are not deterministic, and it is often not sufficient to see an example once. Algorithms update model weights through repetition, so how often an interaction is present in the training data and not just whether it is present may be impactful to the learned model. Training AI models is resource intensive, and running time is measured in terms of size (number of samples) and epochs (number of iterations) through the training data. Thus, there is a trade-off between having sufficient redundant coverage of interactions within the training data to enable learning while minimizing the size of the training data overall for efficiency. Given the nature of statistical learning as reinforcing through repetition and the limitation on number of samples in a training dataset, how those samples are distributed across interactions becomes important to measure.

Previous work on combinatorial frequency measurement has focused on differences between classes for explainable AI and evaluations of training or test data adequacy. In this work, we consider the relative frequencies of interactions in training datasets and whether imbalance, or skew, in frequency coverage impacts performance. Specifically, we investigate whether

This material is based upon work supported, in part, by the U.S. Department of Defense through the Director Operational Test and Evaluation (DOT&E) under Contract HQ0034-24-D-0023. The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology. Any views, opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense nor DOT&E.

Disclaimer: Certain equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement of any product or service by NIST, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

Corresponding author: lanus@vt.edu.

a model that is trained on data with balanced frequency coverage performs better overall on the same test dataset than a model trained on data with skewed frequency coverage.

The rest of the paper is organized as follows. Background on combinatorial testing applied to software and AI is in §II. §III presents the method behind the coverage frequency measure and analysis developed here as well as the data and learning algorithms comprising the experimental setup. Results from the experiments as well as implications for practice are in §IV, while conclusions and future directions are in §V.

II. Background

Combinatorial coverage concerns measuring the degree to which t-way interactions are included in a dataset [2], [3]. Measures may include basic inclusion/exclusion of interactions [1], differences between datasets [2], and differences in frequency of interaction occurrence [4].

A. Combinatorial Coverage in Software Testing

Computing the combinatorial coverage of a test set for a system requires defining an input parameter model of the system under test. Let k be the number of *parameters* in the system. There are $\binom{k}{t}$ ways to select a t-way combination of parameters where t is called the *strength*. A *t-way interaction*, also called a *value combination*, is an assignment of values to the t factors. If the ith parameter in a combination has v_i values, there are $\prod_i^t v_i$ interactions for that combination. The *total t-way coverage* is the proportion of valid t-way interactions that appear at least once in a dataset [1], [5]. Other measures of combinatorial coverage include *simple t-way combination coverage* which measures the proportion of t-way combinations that have all interactions covered and *(p,t)-completeness* which measures the proportion of t-way combinations that have interaction coverage of at least p. An example is shown in Fig. 1, where the total 2-way coverage is the area under the red (rightmost) curve, or roughly 96%. We can also see (1) that 82% of the combinations have all interactions covered, and (2) 98% of the combinations are covered to at least 75% of possible settings.

The various combinatorial coverage measures have direct applications to software testing, and multiple case studies have demonstrated their utility for improving test quality (see for example [6] [7] [8]). Combinatorial coverage of inputs (a static measure) has close connections with dynamic structural code coverage. Interesting properties include levels of combinatorial coverage ensured by some structural coverage criteria [1], and conversely, structural coverage measures ensured by combinatorial coverage under certain assumptions [9].

B. Combinatorial Coverage in Testing AI

AI models are built by training a learning algorithm on samples within a *training dataset*, which should match the environment where the model will be used. Performance on a *validation dataset* is used to estimate model performance on unseen samples typically drawn from the same distribution as the training data to tune learning algorithm hyper-parameters

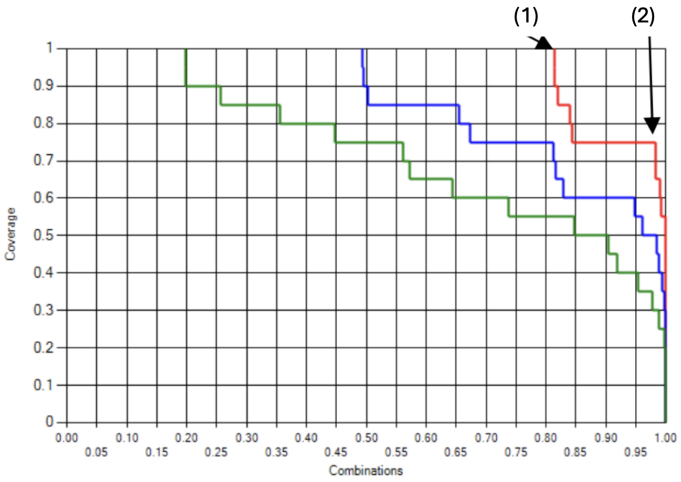


Fig. 1: Basic Combinatorial Coverage Example

or to determine when training should stop. A trained model is evaluated by making predictions or *inferences* on a *test dataset* and comparing these outputs to the labels of the test oracle to measure the model's ability to generalize to new samples. Typically, test datasets are drawn from the same distribution as the training and validation data. Combinatorial coverage provides measures of training and test data distributions that can supplement conventional statistical measures.

Applying combinatorial coverage for testing the AI input space also requires constructing an input parameter model where k is the number of *features* in the input space or *universe* and feature i has v_i discretized values. The *combinatorial coverage* of a dataset is defined as the proportion of valid t-way feature interactions of the universe that are present in a dataset [2]. *Set difference combinatorial coverage (SDCC)* measures the proportion of t-way interactions present in one dataset that are missing from a second dataset. Published results suggest that coverage tends to correlate with performance. Models exhibit better performance on test sets designed to be closer to the training dataset measured by SDCC [10]. Datasets augmented with samples identified by SDCC designed to cover the gap between a source and target domain result in better transfer learning models than when the same number of samples are chosen from the target domain randomly [11]. A systematic inclusion/exclusion framework based on combinatorial testing builds a constellation of training and test datsets with systematically designed coverage gaps to identify the critical features over which coverage computations should be performed [12]. It also outlines a two-phase algorithm for constructing reasonably balanced test datasets where interactions within a combination appear in approximately the same number of samples. Furthermore, [13] provides a comprehensive overview of how combinatorial coverage can be leveraged to address the various testing challenges encountered across the ML-based system's lifecycle.

```
0.068 of cases, Temperature = B3
0.314 of cases, Humidity = B3
0.011 of cases, Light = B2
0.069 of cases, CO2 = B2
0.284 of cases, HumidityRatio = B4

0.000 of cases, Light,CO2,HumidityRatio = B2,B2,B4
0.001 of cases, Humidity,Light,CO2 = B3,B2,B2
0.001 of cases, Temperature,Light,CO2 = B3,B2,B2
0.001 of cases, Humidity,Light,HumidityRatio = B3,B2,B4
```

Fig. 2: Unoccupied Room Feature Combination Frequency

C. Combinatorial Frequency Coverage and Difference

Combinatorial coverage measures described above consider the presence or absence of an interaction, but they do not consider the number of repetitions of an interaction. Additional information may be gained by measuring the *frequency* of occurrence of interactions, such as the difference in how often an interaction appears in samples belonging to one class versus another. Combination frequency coverage, and differences between coverage, has been applied to explainable AI and to measures of training or test data adequacy.

Explainability. If a particular combination of features is associated with class A but not with class B, then it can be expected to occur in a large fraction of A and be rare or not present among members of B. Thus it can be useful to measure the *differences* in frequency of occurrence of value combinations for one class as compared with another class. This notion has been applied to explainability in AI/ML [14] [15]. Single features typically will not distinguish between two classes, but combinations can identify a class. Taxonomies of animals or plants are obvious examples. Extending this approach to ML problems, we can measure the difference in combination frequency between two classes.

An example is shown in Fig. 2, using sensor readings of several values to determine if a room is occupied. Certain levels of individual readings (top) are present in both occupied and unoccupied rooms. For example, 6.8% of unoccupied rooms have temperature value B3 and 6.9% have CO2 value B2, but 3-way combinations (bottom) including these values are present in occupied rooms, but nearly always absent in unoccupied rooms. A human user thus has an intuitive explanation for concluding that a room is occupied – these sensor reading combinations hardly ever occur in unoccupied rooms – and a quantitative assessment (0.1%) of the likelihood that this conclusion might be incorrect.

Dataset Evaluation. Differences in frequency of value combination occurrence can also be applied in analyzing datasets used in ML [4], where the difference in prevalence of various feature value combinations may be significant for distinguishing between classes. One example of combination frequency differences has been in evaluating the vulnerability of physical unclonable functions (PUFs) to ML attacks [16]. A PUF takes advantage of slight deviations in otherwise identical hardware to produce response bit strings that are unique to a particular device. Because they depend on physical properties, they cannot be algorithmically copied, thus "unclonable." The

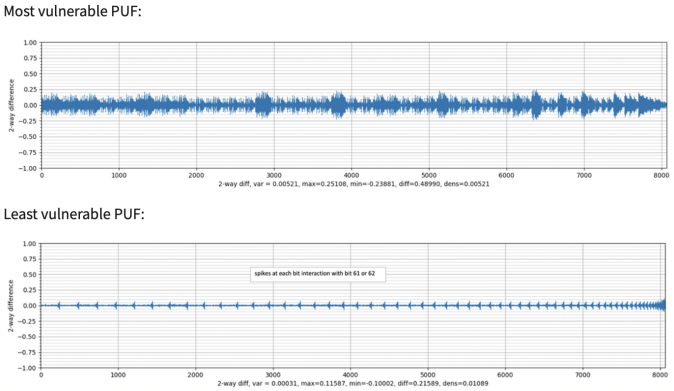


Fig. 3: PUF Data

structure of a PUF is $challenge \longrightarrow response$, mapping a bit string challenge to a single bit response.

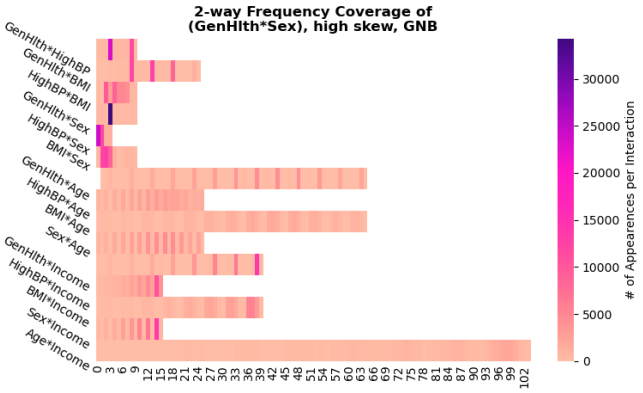
PUFs are susceptible to ML attacks, so it is desirable to analyze their resistance to such attacks. The conventional way of doing this is to analyze the entropy of the responses. Greater entropy suggests better randomization and resistance to ML attacks. A different analysis can be achieved using combination frequency differences, that is, analyzing *challenge* strings that generate a 0 or 1 response, treated as two different classes. Using this approach, it is found that more vulnerable PUFs have greater density in combination frequency differences between challenge strings producing a 0 versus a 1 response [16]. This difference is illustrated visually in Fig. 3. The greater differences indicate some combinations that will have greater influence on the response, thus weakening the PUF.

III. Method and Experimental Setup

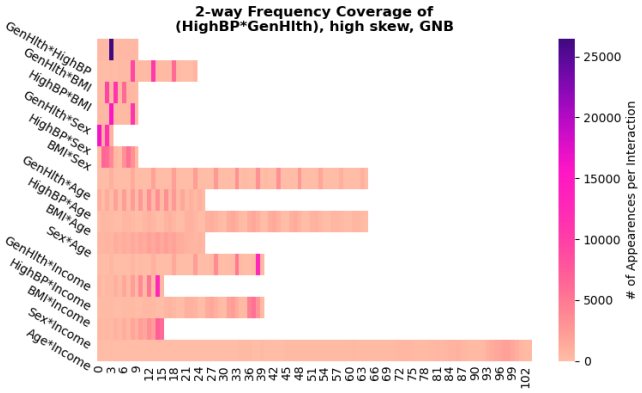
A. Visualizing Frequency Coverage and Performance

One way to visualize the frequency coverage of interactions is to produce a heatmap of absolute counts of each interaction (Figure 4). The scale on the heatmap reflects the maximum number of appearances of any interaction in the dataset, and this is bounded by the dataset size. Thus, directly comparing frequency coverage plots of two datasets with different sizes by looking at colors is misleading, and comparison on the same scale is not possible unless the scale is pre-determined when generating both plots, creating extensibility obstacles. An alternative visualization gives the proportion of the data containing each interaction as absolute count over number of samples producing a score between 0 and 1 (Figure 5).

Most often, ML evaluation results in a single performance metric aggregated across the entire test set or else aggregated per class label. Exploring the correlation of frequency coverage and performance to identify when representation imbalance of interactions may lead to differential performance requires computing performance on a per interaction basis. To do this, rather than aggregating all results from model inference on the test set, we maintain each inference, compare it to the oracle, and produce a per sample accuracy performance

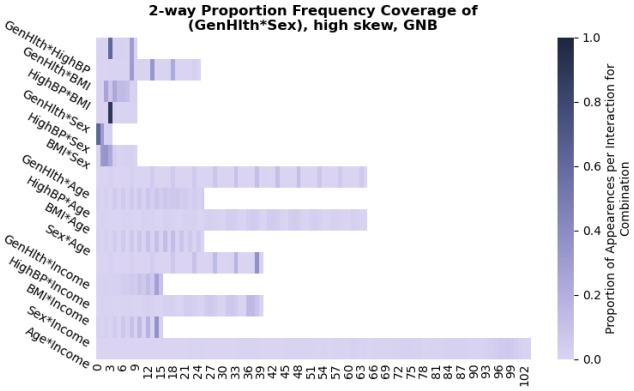


(a) Frequency coverage of 38k sample dataset

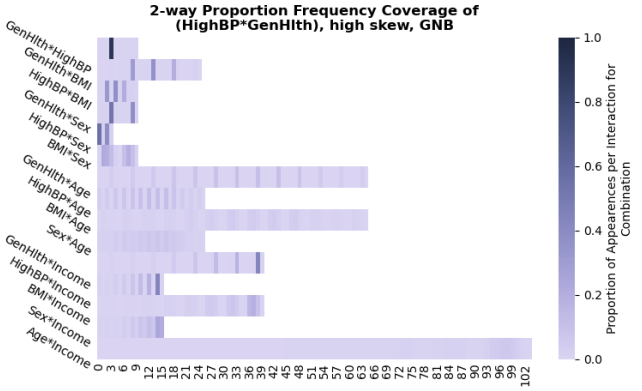


(b) Frequency coverage of 29k sample dataset

Fig. 4: Frequency coverage plots make comparing datasets of differing size challenging



(a) Frequency proportion coverage of 38k sample dataset



(b) Frequency proportion coverage of 29k sample dataset

Fig. 5: Frequency proportion coverage plots show proportion of dataset on [0,1] scale

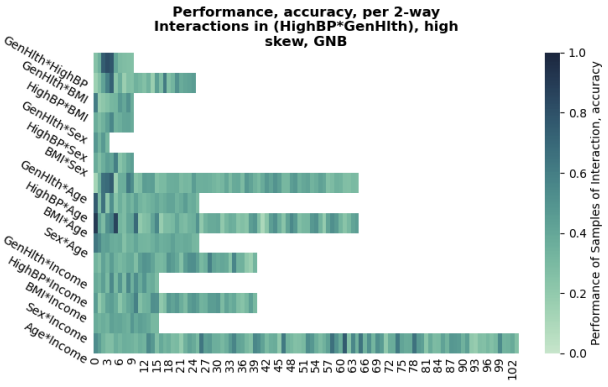


Fig. 6: Performance by interaction for 29k sample dataset

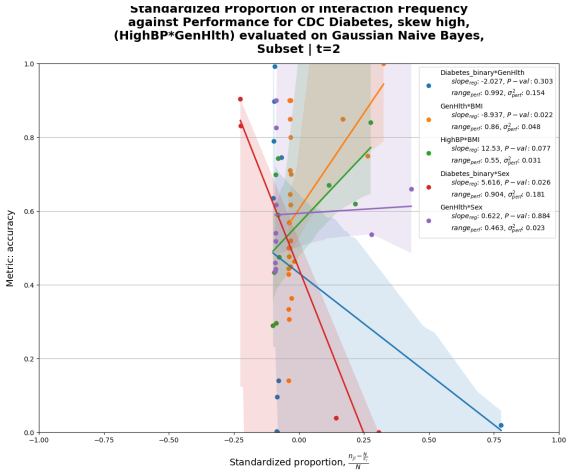


Fig. 7: Performance by standardized frequency coverage plot

score. Then, for each t-way interaction, the scores of each sample that contain that interaction are aggregated to produce an accuracy performance per interaction score. Plotting this

score on a heatmap with the same structure as the frequency coverage heatmap enables visual analysis of correlations between coverage and performance (Figure 6). A strength of this approach is that the interactions within a combination can be indexed to trace coverage and performance differences back to specific interactions; however, this requires a lookup table to retrieve the feature values in the interactions. As is true for all of the heatmaps, as the number of combinations and interactions increases, the visual analysis of heatmaps for overall trends becomes difficult.

An alternative approach is to plot frequency coverage on an x-axis, performance on a y-axis, and fit a line to look for correlation. Frequency coverage may be given as proportion of data as before. That is, if there are 100 samples and some interaction appears in 20 samples, its x-coordinate is 0.20. A disadvantage of this occurs when including points across different combinations. Different combinations have different numbers of interactions amongst which datapoints are spread. That is, consider that combination $\mathcal{A}$ has two interactions and combination $\mathcal{B}$ has four. When perfectly balanced, interactions for $\mathcal{A}$ will appear 50 times each while interactions for $\mathcal{B}$ will appear 25 times each. It could be confusing to interpret a plot where interactions in combination $\mathcal{B}$ have higher performance yet seemingly lower frequency coverage than interactions in combination $\mathcal{A}$. To capture over- or under-representation, instead we standardize the frequency coverage as a score that reflects distance from balance. Let there be N samples, c_i interactions for combination i, and n_{ij} samples for interaction j within combination i. Then N/c_i is the number of samples expected per interaction for combination i in a perfectly balanced dataset, $n_{ij} - N/c_i$ is the distance from balance for interaction j within combination i, and $(n_{ij} - N/c_i)/N$ standardizes this distance for a dataset of size N.

When perfectly balanced, every interaction appears $n_{ij} = N/c_i$ times in the dataset and thus the score for all interactions is $(N/c_i - N/c_i)/N = 0$. The largest value occurs when one interaction appears in all samples, so $n_{ij} = N$ for the most frequently appearing interaction and $n_{ij'} = 0$ for all others. Then the upper bound for a given combination is $(N - N/c_i)/N = 1 - (1/c_i)$ As c_i increases, so does the largest possible score (e.g., $c_i = 2$ gives 0.5 and $c_i = 100$ gives 0.99) approaching 1 asymptotically, so the upper bound for the score across all combinations is 1. The smallest score occurs when there are no samples for an interaction, $n_{ij} = 0$; the most underrepresented an interaction can be in a dataset is to never appear. Then the lower bound for a given combination is $(0 - N/c_i)/N = 0 - 1/c_i$. As c_i increases, so does the lower bound for that combination (e.g., $c_i = 2$ gives -0.5 and $c_i = 100$ gives -0.01). Then the lowerbound for any combination c_i is -0.5 if we require that a combination have at least two interactions and it is -1 if we make no such assumption. Fig. 9 gives examples of standardized proportion frequency coverage plots and Fig. 7 gives an example of plotting performance against standardized frequency coverage.

B. Experimental Method

Not all features or feature interactions in an ML dataset are equally important to the learned function; some are stronger predictors of a class [17], [18]. We hypothesize that skewed frequency coverage of important features has a greater impact on model performance than non-important feature interactions. SHapley Additive exPlanations (SHAP) identifies features that contribute most to a prediction, or to the uncertainty of a prediction, and is often used in the context of explainable AI [17]. To identify the important features, we use the SHAP Python package over the complete original dataset. We then select a set of features to form the "universe" (input parameter model) that includes a mix of features identified by SHAP as important and unimportant, and focus our analysis on a subset of t-way combinations.

We next produce a balanced universal test set using the algorithm provided in [12]. This test set is used to evaluate all models across all skews and all combinations. We form the training pool by removing the test samples from the original dataset. Next, for a selected t-way combination c with i interactions, we identify the least frequently and most frequently appearing interactions within this combination in the training pool; call the counts of these interactions n and m, respectively. Next, we build four training datasets – baseline, low skew, moderate skew, and high skew – each containing approximately ni samples by selecting samples from the training pool as follows:

- Baseline: randomly select ni samples resulting in a dataset that has frequency coverage that is similar to the original dataset.
- Low skew: randomly select n samples containing each of the i interactions for the combination c such that each interaction has frequency coverage $x_{low} = \frac{1}{i}$ resulting in a dataset that is balanced for the combination c.
- High skew: randomly select $\max(x_{high}ni, m)$ samples with x_{high} close to 1 from the most frequently appearing interaction and $\frac{(ni - \max(x_{high}ni, m))}{i-1}$ samples from each of the other $i-1$ interactions, resulting in a dataset where the most frequently appearing interaction has frequency coverage x_{high}. The $\max$ function enables continuing when there are not enough samples to reach the intended frequency coverage.
- Moderate skew: randomly select $\max(x_{mod}ni, m)$ samples from the most frequently appearing interaction with $x_{mod} = \frac{x_{low} + x_{high}}{2}$ and $\frac{(ni - \max(x_{mod}ni, m))}{i-1}$ samples from each of the other $i-1$ interactions resulting in a dataset that is moderately skewed towards the most frequently appearing interaction.

Finally, for each learning algorithm and training dataset, we train a model and evaluate its performance on the balanced universal test with recall, r, and $F1$ performance metrics; we use precision, p, to compute $F1$ but we do not report it. Performance metrics are defined for the positive class diabetes = yes as $r = \frac{TP}{TP+FN}$, $p = \frac{TP}{TP+FP}$, and $F1 = \frac{2pr}{p+r}$. TP is the count of true positives, samples with diabetes that the

model predicts to have diabetes, FP is the count of false positives, samples without diabetes that the model predicts to have diabetes, and FN is the count of false negatives, samples with diabetes that the model predicts to not have diabetes. Recall, also known as true positive rate or sensitivity, measures how well a model correctly identifies samples containing the positive class. It is particularly useful in problems like medical diagnosis. Recall is balanced with precision, also called specificity, in the F1 score.

We also compute the expected values for these metrics for three types of random classifiers given the original dataset distribution over labels. These random classifiers make predictions without using any information about the features of a sample and serve as abstractions to compute the performance metrics we expect to observe if inferences are made by chance, providing a lower bound on performance. A Coin Toss classifier simply flips a fair coin and predicts one of the two binary classes; it predicts the positive class with probability 0.50. A Weighted Random classifier predicts the positive class with probability equal to the proportion in the dataset. A Majority Class classifier always predicts the majority class; the positive class is either predicted with probability 1.0 or 0.0. Note that these metrics are dependent on how the positive class is chosen and give different results if the classes are swapped.

To determine the impact of skewing the frequency coverage on a performance metric, we analyze the variation between the minimum and maximum scores across three intentional skew levels (low, moderate, and high). A difference of 10% or greater indicates a significant impact. Note that, while baseline training dataset results are presented in Table III, IV, V, they are not included in our analysis. We categorize the differences into three categories:

1) Increase (+): the score increases as skew increases; the maximum value is achieved with high skew and the minimum value is achieved with low skew.
2) Decrease (-): the score decreases as skew decreases; the maximum value is achieved with low skew and the minimum value is achieved with high skew.
3) Nonlinear (N): there is a difference, but it is neither increasing nor decreasing; the moderate skew is either the maximum or the minimum.

Our analysis addresses two research questions:

1) Combination comparison: is there a correlation when skewing for some combinations but not others?
2) Learning algorithm comparison: is there a correlation for some learning algorithms and not others?

To address (1), we report the proportion of learning algorithms that are impacted by skew for each of the combinations. To address (2), for each learning algorithm, we report the proportion of combinations that impacts the learning algorithm.

C. Dataset

The Diabetes Health Indicators Dataset by the Centers for Disease Control [19] is a tabular dataset containing responses from a survey conducted by the Behavioral Risk Factor Surveillance System. 253,680 samples span a mix of 21 binary, categorically encoded, and continuous features. Feature values are obtained from respondents' answers to questions about health factors indicating the presence of diabetes and pre-diabetes. These questions vary in specificity and subjectivity, ranging from a body mass index (BMI) value obtained from height and weight to the number of days in the past month that the respondent felt mentally healthy.

TABLE I: Universe of CDC Diabetes dataset defined over features Sex, HighBP, GenHlth, BMI, Income

Sex	HighBP	GenHlth	BMI	Income
Female = 0	No = 0	5	[0,18.5)	'$<$ \$10k' = 1
Male = 1	Yes = 1	4	[18.5, 25)	2
–	–	3	[25, 30)	3
–	–	2	[30, 40)	4
–	–	1	[40, 100]	'$<$ \$35k' = 5
–	–	–	–	6
–	–	–	–	7
–	–	–	–	'$\geq$ \$75k' = 8

While we fit and evaluate the ML models on the entire feature space of the dataset, we select only a subset of the features for experimentation with combinatorial frequency coverage. The selected features include some identified as important by SHAP, specifically General Health Score (GenHlth), High Blood Pressure (HighBP), and BMI, as well as some identified as unlikely to be important, specifically Age, Sex, and Income. We also allow combinations to include the class label, diabetes or no diabetes. We discretize the BMI continuous feature as required for combinatorial testing using the CDC categories [20]. The universe of features and feature values is in Table I.

The universal test set was constructed to be balanced across all 2-way combinations of features in the universe given in Table I resulting in a test set size of 1,514 samples. The high skew frequency coverage goal was set to $x_{high} = 0.90$. In Table III, we report the combinations considered along with the most frequently appearing interaction, the training dataset size, and the count and training data frequency coverage per skew for the most frequently appearing interaction for the targeted combination. We initially included Age as a feature with 13 values, but this led to maximum x_{high} values of 0.26 and 0.49 for interactions of Age with Sex and Income, respectively, preventing meaningful comparison across combinations.

D. Learning Algorithms

Not all learning algorithms may be equally impacted by differences in frequency coverage within training data due to differences in how the learning algorithm represents the data via the model. To counter the possibility that results are algorithm-specific, we evaluate the performance of models from multiple learning algorithms on the same test set. After constructing the training data for each frequency coverage skew level, we train models with five learning algorithms from the scikit-learn Python package [21]. As the purpose of this study is not to achieve optimal model performance but rather to understand how changes in frequency coverage of

interactions impact model performance, we train all models within a learning algorithm with the same hyper-parameters. We leverage the scikit-learn GridSearchCV package which searches exhaustively over a set of hyper-parameters and performs k-fold cross validation [22]. Given time constraints, we set the training data size to 25,000 samples with five folds resulting in the following learning algorithm hyper-parameter settings:

- Gaussian Naive Bayes (GNB): var_smoothing=4.328761281083062e-05
- Logistic Regression (LR): solver='sag', C=1, penalty=l2
- Random Forest (RF): criterion=log_loss, max_depth=10, max_features=sqrt, n_estimators=100
- k-Nearest Neighbors (KNN): algorithm=auto, leaf_size=50, n_neighbors=5, weights=uniform
- Support Vector Machine (SVM): C=10, decision_function_shape=ovo, kernel=poly

IV. Results and Discussion

A. Experimental Results

Table II reports performance metrics for the three random classifiers given that the positive class, Diabetes=yes, appears in 14% of samples in the original dataset. Table III reports, for each combination considered, the most frequently appearing interaction within the combination as well as the constructed training dataset size for each skew, the count m and frequency coverage of the most frequently appearing interaction in the training dataset. For all, $x_{high} = 0.90$, but x_{low} by definition is dependent on the number of interactions within the combination, reflecting equal frequency coverage across all interactions. Performance metrics for each skew for each learning algorithm for each combination are reported with recall in Table IV and $F1$ in Table V. For each combination, the proportion of learning algorithms impacted by skew for recall is in Table VI and for $F1$ is in Table VII. For each learning algorithm, the proportion of combinations impacted by skew for recall is in Table VIII and for $F1$ is in Table IX.

The combination for which the most learning algorithms are impacted by skew is Diabetes x BMI, followed by Diabetes x Income and BMI x GenHlth, and then HighBP x GenHlth. For all combinations, at least 40% of learning algorithms are impacted for recall and at least 20% for $F1$. We do not have enough evidence that skew in SHAP-identified important combinations has a stronger impact on model performance. Specifically, skew in Diabetes x BMI (label and important) impacts the same number of learning algorithms as skew in Diabetes x Income (label and not important) for recall, but for $F1$, there is a strong difference (100% vs. 40%). Skew in combinations Sex x Income, Diabetes x Sex, and Diabetes x HighBP all result in the same proportion of learning algorithms impacted for both recall and $F1$. Bias in the label does not always have the strongest impact; compare the impact of Diabetes x Income and Diabetes x BMI versus Diabetes x HighBP or Diabetes x Sex.

TABLE II: Expected recall, F1 for random classifier

	Coin Toss	Weighted Random	Majority Class
Precision	0.14	0.14	0.00
Recall	0.50	0.14	0.00
F1	0.22	0.14	0.00

The learning algorithm impacted for the most combinations for recall is LR followed by GNB and KNN. For $F1$, these three are equally impacted. All learning algorithms are impacted by skew in at least one combination for both metrics.

Models generally perform better than the random classifiers on $F1$ except RF trained on high skew for BMI x GenHlth and Diabetes x BMI. SVM performs worse than coin toss for recall for all skews and the baseline for BMI x GenHlth and all but the low skew for Diabetes x BMI. KNN rarely performs better than the coin toss classifier for recall, and RF performs better less than half of the time. Any conclusions about resilience to skew in the training data should be tempered by considering whether the learning algorithm ever produces reasonable models on this dataset.

Fig. 8 shows the proportion of learning algorithms that are impacted by skew for each training data size. While fewer learning algorithms are impacted for the largest training datasets, there is not a clear correlation. That is, adding more data is not necessarily protective against performance changes caused by skew and using less data does not always cause vulnerability to changes due to skew. KNN exhibits generally poor performance across most skews and combinations, regardless of the training dataset size.

The hypothesis that performance decreases with an increase in skew not always supported; it often increases. We skew on the most frequently appearing interaction, so depending on the original frequency coverage, creating a balanced dataset is biasing away from the original distribution. However, the frequency coverage of the most frequently appearing interaction in the baseline dataset is below the frequency coverage in the moderate skew dataset in all combinations with frequency

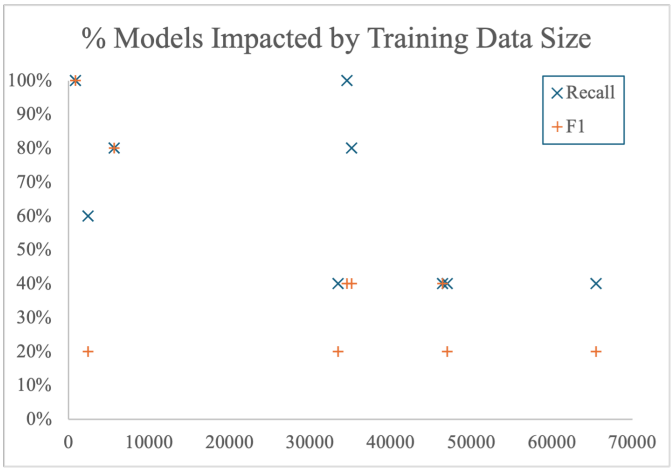


Fig. 8: Training dataset size against % algorithms impacted

TABLE III: Training Dataset Sizes and Skew Proportions

most frequent	BMI*GenHlth [25, 30), Very Good				Diabetes*BMI No, [25,30)				Diabetes*HBP No, No			
skew	B	L	M	H	B	L	M	H	B	L	M	H
training size	5675	5675	5691	5683	890	890	895	891	33500	33500	33500	33501
m	801	227	2667	5107	300	89	445	801	18016	8375	19262	30150
proportion	0.14	0.04	0.47	0.9	0.34	0.1	0.5	0.9	0.54	0.25	0.57	0.9
	Diabetes*Income				Diabetes*Sex				GenHlth*Sex			
most frequent	No, 75k+				No, Female				Very Good, Female			
skew	B	L	M	H	B	L	M	H	B	L	M	H
training size	34560	34560	34572	34569	65512	65512	65512	65512	46460	46460	46468	46467
m	11500	2160	16632	31104	32050	16378	37669	58960	9294	4646	23230	41814
proportion	0.33	0.06	0.48	0.9	0.49	0.25	0.57	0.9	0.2	0.1	0.5	0.9
	HBP*GenHlth				Sex*Income							
most frequent	No, Very Good				Male, $75k+							
skew	B	L	M	H	B	L	M	H				
training size	35160	35160	35166	35163	47024	47024	47035	47031				
m	7899	3516	17580	31644	12924	2939	22630	42321				
proportion	0.22	0.1	0.5	0.9	0.27	0.06	0.48	0.9				

TABLE IV: Recall scores for all models, combinations, and skews

	BMI*GenHlth					Diabetes*BMI					Diabetes*HBP				
skew	B	L	M	H	δ	B	L	M	H	δ	B	L	M	H	δ
GNB	0.64	0.48	0.58	0.84	0.36	0.64	0.65	0.63	0.96	0.33	0.64	0.76	0.74	0.80	0.06
LR	0.69	0.59	0.69	0.80	0.21	0.72	0.58	0.55	0.38	0.20	0.72	0.67	0.77	0.84	0.17
RF	0.25	0.28	0.25	0.06	0.21	0.14	0.64	0.36	0.07	0.58	0.47	0.64	0.65	0.63	0.03
KNN	0.17	0.17	0.15	0.07	0.10	0.23	0.64	0.36	0.04	0.60	0.17	0.67	0.49	0.21	0.46
SVM	0.36	0.39	0.37	0.35	0.03	0.39	0.53	0.36	0.26	0.27	0.53	0.62	0.67	0.62	0.05
	Diabetes*Income					Diabetes*Sex					GenHlth*Sex				
skew	B	L	M	H	δ	B	L	M	H	δ	B	L	M	H	δ
GNB	0.63	0.67	0.72	0.84	0.18	0.63	0.74	0.71	0.80	0.09	0.63	0.53	0.63	0.87	0.34
LR	0.72	0.59	0.74	0.89	0.30	0.70	0.71	0.77	0.83	0.12	0.71	0.58	0.71	0.83	0.25
RF	0.47	0.53	0.67	0.70	0.17	0.53	0.65	0.68	0.62	0.06	0.50	0.43	0.53	0.49	0.09
KNN	0.19	0.58	0.49	0.34	0.25	0.18	0.68	0.52	0.17	0.51	0.15	0.20	0.19	0.14	0.05
SVM	0.58	0.58	0.68	0.64	0.10	0.65	0.71	0.75	0.67	0.07	0.62	0.57	0.63	0.57	0.07
	HBP*GenHlth					Sex*Income									
skew	B	L	M	H	δ	B	L	M	H	δ					
GNB	0.64	0.54	0.64	0.86	0.32	0.63	0.53	0.62	0.84	0.30					
LR	0.72	0.61	0.74	0.89	0.28	0.71	0.62	0.68	0.75	0.13					
RF	0.50	0.42	0.53	0.43	0.11	0.49	0.45	0.50	0.44	0.06					
KNN	0.18	0.16	0.17	0.13	0.03	0.19	0.17	0.16	0.18	0.02					
SVM	0.59	0.57	0.60	0.47	0.13	0.60	0.58	0.61	0.56	0.05					

TABLE V: F1 scores for all models, combinations, and skews

	BMI*GenHlth					Diabetes*BMI					Diabetes*HBP				
skew	B	L	M	H	δ	B	L	M	H	δ	B	L	M	H	δ
GNB	0.61	0.52	0.58	0.69	0.17	0.61	0.62	0.60	0.75	0.14	0.61	0.67	0.66	0.67	0.02
LR	0.63	0.57	0.63	0.68	0.11	0.65	0.58	0.56	0.46	0.12	0.65	0.63	0.67	0.69	0.06
RF	0.36	0.37	0.35	0.12	0.26	0.22	0.62	0.43	0.11	0.51	0.54	0.63	0.64	0.62	0.01
KNN	0.27	0.26	0.24	0.13	0.13	0.32	0.62	0.44	0.07	0.55	0.27	0.64	0.53	0.31	0.32
SVM	0.46	0.46	0.45	0.44	0.02	0.47	0.55	0.43	0.34	0.21	0.58	0.62	0.63	0.60	0.03
	Diabetes*Income					Diabetes*Sex					GenHlth*Sex				
skew	B	L	M	H	δ	B	L	M	H	δ	B	L	M	H	δ
GNB	0.60	0.63	0.65	0.69	0.06	0.60	0.66	0.64	0.67	0.03	0.61	0.55	0.60	0.71	0.16
LR	0.65	0.59	0.65	0.71	0.12	0.64	0.64	0.66	0.68	0.04	0.64	0.58	0.64	0.69	0.12
RF	0.54	0.56	0.63	0.66	0.09	0.57	0.63	0.63	0.60	0.03	0.55	0.50	0.57	0.57	0.07
KNN	0.29	0.59	0.53	0.42	0.18	0.27	0.63	0.55	0.26	0.37	0.23	0.30	0.28	0.23	0.07
SVM	0.60	0.60	0.64	0.62	0.04	0.63	0.65	0.66	0.63	0.03	0.62	0.59	0.63	0.61	0.04
	HBP*GenHlth					Sex*Income									
GNB	0.61	0.55	0.61	0.69	0.14	0.60	0.55	0.60	0.70	0.14					
LR	0.65	0.59	0.66	0.72	0.13	0.64	0.60	0.63	0.66	0.06					
RF	0.55	0.50	0.57	0.53	0.07	0.55	0.51	0.55	0.53	0.04					
KNN	0.28	0.24	0.26	0.22	0.04	0.29	0.26	0.25	0.27	0.03					
SVM	0.60	0.59	0.61	0.54	0.06	0.60	0.59	0.60	0.60	0.01					

TABLE VI: Combination comparison: proportion of learning algorithms per combination impacted for recall

	BMI*GenHlth	Diabetes*BMI	Diabetes*HighBP	Diabetes*Income	Diabetes*Sex	GenHlth*Sex	HighBP*GenHlth	Sex*Income
R+	0.4	0.2	0.2	0.6	0.2	0.4	0.4	0.4
R-	0.4	0.8	0.2	0.2	0.2	0.0	0.0	0.0
NR	0.0	0.0	0.0	0.2	0.0	0.0	0.4	0.0
Total	0.8	1.0	0.4	1.0	0.4	0.4	0.8	0.4

TABLE VII: Combination comparison: proportion of learning algorithms per combination impacted for impacted F1

	BMI*GenHlth	Diabetes*BMI	Diabetes*HighBP	Diabetes*Income	Diabetes*Sex	GenHlth*Sex	HighBP*GenHlth	Sex*Income
F1+	0.4	0.2	0.0	0.2	0.0	0.4	0.4	0.2
F1-	0.4	0.8	0.2	0.2	0.2	0.0	0.0	0.0
NF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Total	0.8	1.0	0.2	0.4	0.2	0.4	0.4	0.2

TABLE VIII: Learning algorithm comparison on recall

	GNB	LR	RF	KNN	SVM
R+	0.75	0.875	0.125	0	0
R-	0	0.125	0.25	0.625	0.125
NR	0	0	0.125	0	0.25
Total	0.75	1	0.5	0.625	0.375

TABLE IX: Learning algorithm comparison on F1

	GNB	LR	RF	KNN	SVM
F1+	0.625	0.5	0	0	0
F1-	0	0.125	0.25	0.625	0.125
NF	0	0	0	0	0
Total	0.625	0.625	0.25	0.625	0.125

coverage more similar to low than high skew (Fig. 9).

B. Limitations

We used random sampling throughout our experiments: in constructing the universal balanced test set, random selection of samples for training datasets given coverage constraints, and randomness in the model training process. This randomness shall introduce result variability. Experimentation was broad – eight combinations, five learning algorithms, four training datasets skews – instead of deep with replicates to account for variability due to randomness.

The analysis requires setting a threshold to determine when a change in recall exists across the skews. The threshold of 10% ensures that the change in performance is not minor, as this change in performance is typically considered impactful. Using 9% or 12% as the threshold leads to including or excluding additional learning algorithms as impacted.

The analysis is performed only for $t = 2$. Several combinations that result in large changes in performance or changes for many learning algorithms subsume the same feature (e.g., GenHlth or BMI). The analysis may attribute to higher strength interactions an effect explainable by a single feature.

V. Conclusion and Future work

This study confirms that changes in the frequency coverage of interactions within training datasets often impact model performance. Skew in combinations identified by feature importance methods or those involving the class label may not necessarily be more impactful than skew in other combinations, suggesting that coverage may be required across all feature combinations, not just for important ones. Surprisingly, skew may lead to performance improvements, likely because it reflects the population data distribution more closely than expected. Furthermore, adding more data may not mitigate skew effects, suggesting that representative frequency coverage in training data is likely more important than quantity. The combinatorial coverage measure yields a single quantitative value which can be used to compare different datasets, and set difference combinatorial coverage can be used to compare the distance between two datasets. The method here computes frequency coverage for each interaction, producing a distribution of coverage values for a dataset. This is useful for looking within a dataset to understand the balance of interactions, but it is not clear how to compare datasets using their frequency coverage values. Measures that quantitatively describe the distribution of frequencies within a combination and across combinations within a dataset are needed to compare frequency coverage at the dataset level.

This study was limited to selecting some 2-way feature combinations. A broader study should extend the analysis to other features in this dataset or other datasets, as well as other values of t and with replicates to account for randomness.

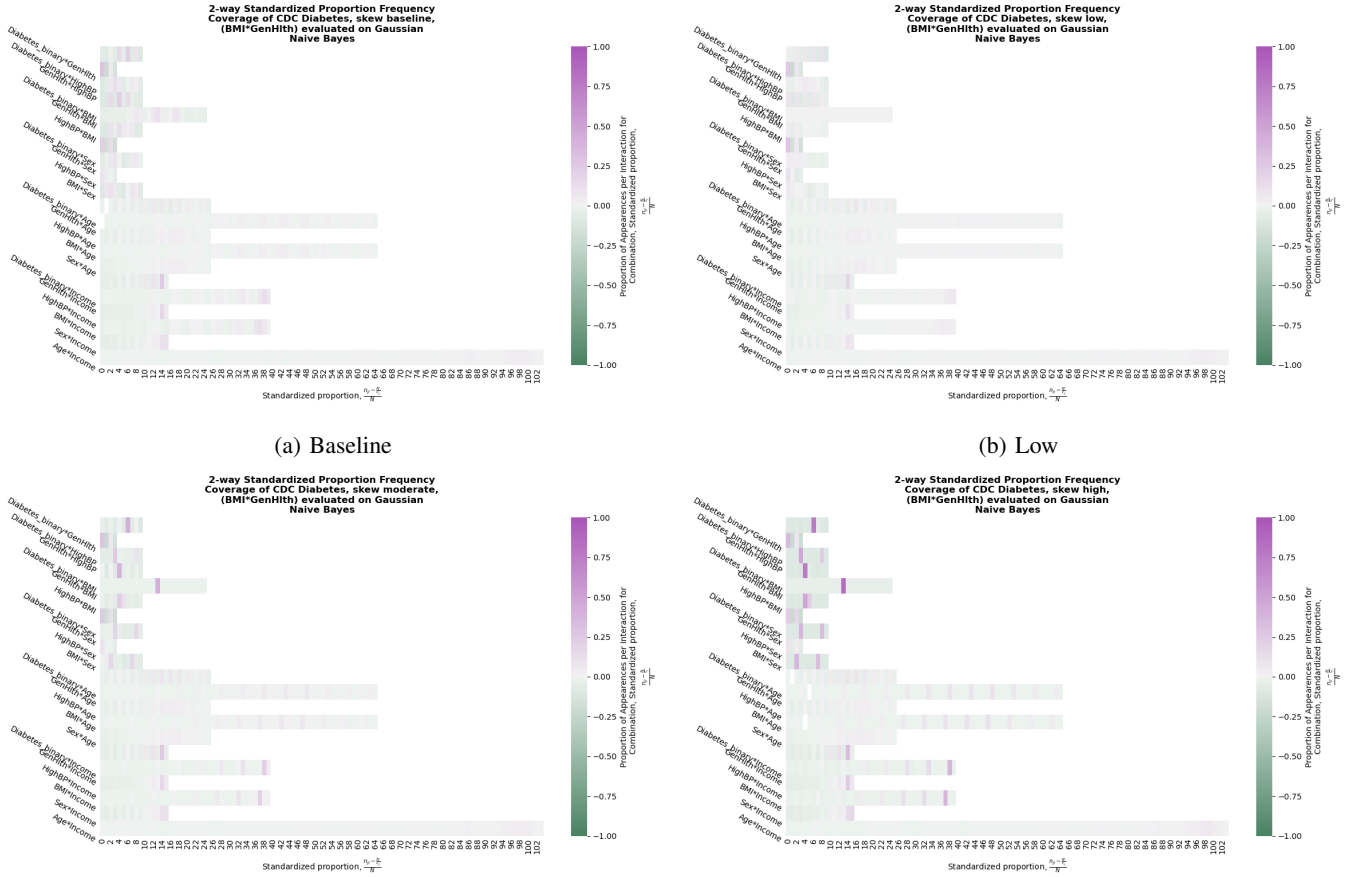
Additional analysis is needed to determine if the current method can provide explanations for why performance differs. A finer-grained analysis investigating performance changes by interaction may indicate how biasing interactions in the data affects the models performance on each interaction separately.

References

[1] D. R. Kuhn, I. Dominguez Mendoza, R. N. Kacker, and Y. Lei, "Combinatorial coverage measurement concepts and applications," in *2013 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2013, pp. 352–361.

[2] E. Lanus, L. J. Freeman, D. R. Kuhn, and R. N. Kacker, "Combinatorial testing metrics for machine learning," in *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2021, pp. 81–84.

[3] M. S. Raunak, D. R. Kuhn, R. N. Kacker, and Y. Lei, "Ensuring reliability through combinatorial coverage measures," *IEEE Reliability Magazine*, vol. 1, no. 2, pp. 20–26, 2024.



(a) Baseline

(b) Low

(c) Moderate

(d) High

Fig. 9: Frequency coverage across training datasets for combination BMI x GenHlth

[4] R. Kuhn, M. Raunak, and R. Kacker, "Combination frequency differencing," National Institute of Standards and Technology, US Department of Commerce, Tech. Rep., 2021.

[5] M. S. Raunak, D. R. Kuhn, R. N. Kacker, and J. Y. Lei, "Combinatorial testing for building reliable systems," *IEEE Reliability Magazine*, vol. 1, no. 1, pp. 15–19, 2024.

[6] M. Ozcan, "Applications of practical combinatorial testing methods at siemens industry inc., building technologies division," in *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2017, pp. 208–215.

[7] R. Smith, D. Jarman, J. Bellows, R. Kuhn, R. Kacker, and D. Simos, "Measuring combinatorial coverage at adobe," in *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2019, pp. 194–197.

[8] M. Fifo, E. Enoiu, and W. Afzal, "On measuring combinatorial coverage of manually created test cases for industrial software," in *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2019, pp. 264–267.

[9] D. R. Kuhn, R. N. Kacker, and Y. Lei, "Measuring and specifying combinatorial coverage of test input configurations," *Innovations in systems and software engineering*, vol. 12, pp. 249–261, 2016.

[10] T. Cody, E. Lanus, D. D. Doyle, and L. Freeman, "Systematic training and testing for machine learning using combinatorial interaction testing," in *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2022, pp. 102–109.

[11] S. F. Ahamed, P. Aggarwal, S. Shetty, E. Lanus, and L. J. Freeman, "ATTL: An automated targeted transfer learning with deep neural networks," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–7.

[12] E. Lanus, B. Lee, L. Pol, D. Sobien, J. Kauffman, and L. J. Freeman, "Coverage for identifying critical metadata in machine learning operating envelopes," in *2024 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2024, pp. 217–226.

[13] J. Chandrasekaran, E. Lanus, T. Cody, L. J. Freeman, R. N. Kacker, M. S. Raunak, and D. R. Kuhn, "Leveraging combinatorial coverage in the machine learning product lifecycle," *Computer*, vol. 57, no. 7, pp. 16–26, 2024.

[14] D. R. Kuhn, R. N. Kacker, Y. Lei, and D. E. Simos, "Combinatorial methods for explainable AI," in *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2020, pp. 167–170.

[15] L. Kampel, D. E. Simos, D. R. Kuhn, and R. N. Kacker, "An exploration of combinatorial testing-based approaches to fault localization for explainable ai," *Annals of Mathematics and Artificial Intelligence*, pp. 1–14, 2022.

[16] D. R. Kuhn, M. S. Raunak, C. Prado, V. C. Patil, and R. N. Kacker, "Combination frequency differencing for identifying design weaknesses in physical unclonable functions," in *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2022, pp. 110–117.

[17] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[18] K. Khadka, S. Shree, Y. Lei, R. N. Kacker, and D. R. Kuhn, "Assessing the degree of feature interactions that determine a model prediction," in *2024 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2024, pp. 180–187.

[19] CDC, "Diabetes health indicators," 2023.

[20] C. for Disease Control, "Adult BMI categories," 2024.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[22] Scikit-learn, "Gridsearchcv - scikit learn 1.6.1 documentation."