Effectiveness of dataset reduction in testing machine learning algorithms

Jaganmohan Chandrasekaran, Huadong Feng
Department of Computer Science & Engineering
The University of Texas at Arlington
Arlington, USA
{jaganmohan.chandrasekaran, huadong.feng}@mavs.uta.edu

Yu Lei
Department of Computer Science & Engineering
The University of Texas at Arlington
Arlington, USA
ylei@cse.uta.edu

Raghu Kacker, D. Richard Kuhn
Information Technology Lab
National Institute of Standards and Technology
Gaithersburg, USA
{raghu.kacker, d.khun}@nist.gov

***Abstract*— Many machine learning algorithms examine large amounts of data to discover insights from hidden patterns. Testing these algorithms can be expensive and time-consuming. There is a need to speed up the testing process, especially in an agile development process, where testing is frequently performed. One approach is to replace big datasets with smaller datasets produced by random sampling. In this paper, we report a set of experiments that are designed to evaluate the effectiveness of using reduced datasets produced by random sampling for testing machine learning algorithms. In our experiments, we use as subject programs four supervised learning algorithms from the Waikato Environment for Knowledge Analysis (WEKA). We identify five datasets from Kaggle.com to run with the four learning algorithms. For each dataset, we generate reduced datasets of different sizes using two random sampling strategies, i.e., pure random and stratified random sampling. We execute our subject programs with the original and the reduced datasets, and measure test effectiveness using branch and mutation coverage. Our results indicate that in most cases, reduced datasets of even very small sizes can achieve the same or similar coverage achieved by the original dataset. Furthermore, our results indicate that reduced datasets produced by the two sample strategies do not differ significantly, and branch coverage correlates with mutation coverage.**

Keywords*— *Testing classifiers, Random sampling, Reduced datasets, Testing machine learning, Branch coverage, Software testing.

I. INTRODUCTION

Many machine learning algorithms examine large amounts of data to discover insights from hidden patterns. Given the nature of machine learning algorithms, testing can be expensive and time-consuming as each test case may have longer execution time compared to the testing of traditional applications. There is a need to speed up the testing process, especially in an agile development process, where testing is frequently performed. One approach is to replace high volume test datasets with smaller datasets produced by random sampling. One natural question to ask about this approach is the following: How does a reduced dataset compare to the original dataset in terms of effectiveness from a testing perspective?

In this paper, we investigate the effectiveness of using reduced datasets for testing machine learning algorithms. We measure test effectiveness using both branch coverage and mutation coverage. In our study, we use four supervised learning algorithms from the Waikato Environment for Knowledge Analysis (WEKA), which is a widely used machine learning workbench written in Java [1], as our subject programs. We identify five datasets, each of which represents a different application domain, from Kaggle.com to run with these algorithms. Kaggle.com is an online data science community that maintains a repository of public datasets.

After we identify subject programs and datasets, we first execute each subject program with each of the five datasets and measure test effectiveness in terms of branch coverage and mutation coverage. Second, we create two groups of reduced datasets. The first group is generated using pure random sampling, i.e., in a purely random manner. The second group is generated using stratified random sampling, i.e., in a random manner that maintains the class distribution. In another word, a reduced dataset produced by stratified random sampling has the same class distribution as the original dataset. In the rest of the paper, we will refer to pure random sampling as random sampling and stratified random sampling as stratified sampling. Third, we execute the reduced datasets with subject programs and measure branch and mutation coverage. Finally, we compare the coverage results achieved by the reduced datasets to the coverage results achieved by the original datasets.

The major findings from our experiments are summarized as follows:

- In most cases, reduced datasets of even very small sizes achieve coverage identical or similar to the original datasets. In our experiments, the original datasets have the number of instances ranging from 142,193 to 999,999. The reduced datasets are of four sizes, i.e., 100, 200, 400, and 800, which are a fraction of the original dataset size. However, 522 out of 800 reduced datasets achieved the same coverage as the original datasets. Also, 112 out of 800 reduced datasets achieved more than 90% of the coverage achieved by the original datasets.
- One might expect that stratified sampling can be more effective than random sampling. However, in our experiments, the coverage results of the reduced datasets produced by the two sampling strategies are very similar. In particular, 628 out of 800 reduced datasets produced by the two sampling strategies achieved the same coverage. It is interesting to note that in several cases, random sampling achieved higher coverage than stratified sampling. The reason is that when the sample size is small, and when the dataset is skewed in terms of class distribution, stratified sampling may produce no instances for a

978-1-7281-6984-2/20/$31.00 ©2020 IEEE
DOI 10.1109/AITEST49225.2020.00027

particular class, which could significantly reduce coverage.

- In most cases, branch coverage correlates with mutation coverage. Since mutation testing is quite expensive to perform, this suggests that branch coverage could be used as a practical alternative in place of mutation coverage for testing machine learning algorithms.

The rest of the paper is organized as follows. In Section II we present the design of our experiments, including the research questions, subject programs, datasets and metrics used in our experiments, and discussion about the generation of reduced datasets. Section III presents the results of our experiments, including branch and mutation coverage results for original and reduced datasets as well as implications of these results. Section IV discusses potential threats to validity, including both internal and external threats. Section V reviews existing work that is related to ours. Section VI provides conclusion remarks and a few directions for the future work.

II. Experimental Design

In this section, we present how we design our experiment, including the research questions, the selection of subject programs and datasets, the sampling approaches used to generate reduced datasets, and the metrics used to measure the effectiveness of the dataset executions.

A. Research Questions

Our main objective is to investigate the effectiveness of using a reduced dataset (in terms of volume, i.e., number of instances in a dataset) to test machine learning algorithms. We formulate the following research questions:

- How effective is it to test machine learning algorithms using reduced datasets, in comparison with the original datasets?
- How do the two sampling strategies, i.e., random sampling and stratified sampling, compare to each other?
- In testing machine learning algorithms, can branch coverage be used as a substitute for mutation coverage?

B. Subject Programs

Waikato Environment for Knowledge Analysis (WEKA) is a machine learning workbench developed by University of Waikato. WEKA has a collection of supervised and unsupervised algorithms implemented in Java. Using WEKA, a user can perform tasks such as classification, regression, clustering and association rule mining. Four supervised algorithms from WEKA are used as our subject programs.

In WEKA, classification algorithms are categorized into seven different groups. We select one algorithm from each of the following four groups, *bayes*, *meta*, *rules* and *trees.* When we choose one algorithm from a group, we only consider algorithms that satisfy two conditions: (1) they support datasets with nominal class labels and (2) they generate a *model* at the end of its training phase. When there are multiple algorithms that satisfy the two conditions, we randomly choose one from these algorithms. The reason for condition (1) is that we use WEKA`s built-in filter to generate smaller datasets for stratified sampling. This filter is applicable only to datasets with nominal class labels. The reason for condition (2) is that during mutation testing, we need expected output to determine if a mutant is killed by comparing against the actual output. If an algorithm generates a model, then the model can be used as expected output during mutation testing.

For example, WEKA lists eight algorithms under the *trees* category. However, one of the eight algorithms, M5p, does not work on a nominal class label. Hence we exclude M5P. Similarly, of the remaining seven algorithms, *random forest* works on a nominal class label dataset, but at the end of its training phase, the model is not accessible to the user with default configuration options. Hence, we exclude *random forest*. From the remaining six algorithms, we randomly select j48 as one of our subject algorithms.

Among different categories of classifiers listed in WEKA, we selected four algorithms namely NaiveBayes classifier [27], AdaBoost1 classifier [28], OneR classifier [29] and J48 classifier [30]. Table I lists our subject algorithms and some information about these algorithms, including package/class information, and number of branches and mutants. Each algorithm is executed with its default configuration values (as provided in WEKA) using command line interface (CLI).

Table I also lists information about an algorithm called DecisionStump. Classification accuracy of simple learning algorithms (weak learners), e.g., decision trees, naïve bayes, can be affected by potential bias in the training dataset. Thus, ensemble classifiers are used to improve their classification accuracy. AdaBoost1 belongs to a class of ensemble classifiers (boosting) that help to improve the classification accuracy of weak learners by training them iteratively, with different sets of weights assigned to class labels in each iteration. WEKA's default configuration of *AdaBoost1* implements a meta classifier that improves the accuracy of the model built using *DecisionStump,* a tree-based classifier (weak learner).

Algorithm	Subject Programs	Number of Branches	Number of Mutants
j48	weka.classifiers.trees.j48*	750	3796
NaiveBayes	weka.classifiers.bayes.NaiveBayes.java	203	1075
AdaBoost1	weka.classifiers.meta.AdaBoost1.java	90	491
DecisionStump	weka.classifiers.trees.DecisionStump.java	128	921
OneR	weka.classifiers.rules.OneR.java	88	510

Table I – Information about subject programs

C. Datasets

We identify suitable datasets from Kaggle.com, which provides access to public databases. By default, dataset search results on Kaggle.com are sorted by hotness, a measure indicative of the amount of interests and recency of datasets on their platform [9]. Other methods of sorting include *New, Recently Active, Most Votes, Updated and Relevance.* As Kaggle.com does not release the hotness calculation formula to the public [10], we are not completely clear of how the hotness of datasets is computed. Hence, we

sort the search results by *Most Votes*, which sorts datasets based on the most popular datasets of all time. Then, the results are further filtered with the following two criteria: (a) *size – 10 MB to 1GB* and (b) *File types – CSV*. Next, we inspect each dataset in the order sorted by Kaggle.com and select datasets that require no cleaning and can be executed in WEKA.

We identified five datasets from different application domains, including *AustralianWeather [23], ForestCover [24, 25], Crime [26]*, *SupplyChain* [21] and *VideoGames [22]*. The *ForestCover* dataset is a multi-label classification dataset with seven different class labels. The remaining four datasets consist of binary class labels. Table II lists the datasets and their information.

We selected datasets such that data preprocessing is minimal. No modification was required for *AustralianWeather and SupplyChain* as their respective class labels were nominal by default. The class labels of the remaining three datasets, i.e. *ForestCover, Crime and VideoGames* were converted from numeric to nominal using WEKA's built-in filter.

DATASET	# OF CLASS LABELS	# OF INSTANCES	# OF ATTRIBUTES
ForestCover	7	581,012	55
AustralianWeather	2	142,193	23
Crime	2	284,807	31
SupplyChain	2	580,251	5
VideoGames	2	999,999	56

TABLE II – DATASET INFORMATION

D. Generation of Reduced Datasets

For each original dataset in Table II, two groups of smaller datasets are generated. Group 1 consists of reduced datasets generated using pure random sampling, whereas in Group 2, reduced datasets are generated using stratified sampling. Recall that stratified sampling maintains the overall class distribution of the original datasets. For each group, we generate samples of four different sizes, i.e., 100, 200, 400, 800. Also, in order to reduce variations in random sampling, we generate five samples for each sample size by using different random seeds. Thus, each dataset has 20 samples per group and a total of 40 samples in the two groups.

WEKA provides a set of pre-processing filters that allow users to modify datasets. Reduced datasets in Group 1 (random sampling) are generated using WEKA's pre-processing filter *weka.filters.**unsupervised**.instances.Resample*. Reduced datasets in Group 2 (stratified sampling) are generated using pre-processing filter *weka.filters.**supervised**.instances.Resample*. These filters allow the user to select the sample size, usually specified as a percentage of the original dataset. Note that both filters perform a volumetric reduction, i.e. the number of instances in the dataset is reduced whereas the number of attributes will remain unchanged.

For example, consider a dataset of 100,000 data instances with four class labels, A, B, C and D. Assume that their class distribution is as follows: 30% instances belong to Class A, 40% instances belong to Class B, 10% instances belong to Class C and the remaining 20% belongs to Class D. Generating a smaller dataset with 100 instances using stratified sampling (Group II) will consists of 30 instances belonging to Class A, 40 instances belonging to Class B, 10 instances belonging to Class C and 20 instances belonging to Class D. In contrast, samples generated using random sampling (Group I) does not necessarily maintain the class label distribution.

The *Crime dataset* (284,807 instances) has the following class distribution: 99.82% instances belong to Class 0 (284,315 instances), and 0.18% instances belong to Class 1 (492 instances). When generating a reduced dataset with 800 instances using WEKA's pre-processing filter, it is highly likely that random sampling fails to produce a reduced dataset that include instances in both Class 0 and Class 1. Instead, it is likely that all of the 800 instances belong to Class 0. A developer might face the above said scenario when s/he generates a reduced dataset using random sampling from a class-imbalanced (or skewed) dataset. As a workaround, a developer can create a reduced dataset while preserving the original class distribution. This is our motivation to use two different groups of samples and to investigate their impact in testing supervised learning algorithms. The original datasets and their reduced versions are made publicly available at [32].

E. Metrics

We use both branch coverage and mutation coverage to measure test effectiveness. Branch coverage is recorded using JaCoCo [18]. We choose branch coverage over statement coverage because the former subsumes the latter. We note that logic coverage is stronger than branch coverage. Unfortunately, JaCoCo does not report logic coverage.

Mutation coverage is obtained using PITest (PIT), which is a widely used mutation testing framework [19]. PIT can automatically seed one fault at a time into SUT and execute the mutated code against the unit test(s) specified. We executed each dataset with WEKA's default configuration options and the output (model) is saved in a .txt file (expected output). Then, we used jUnit tests to compare the expected output against the output of each mutated version. If the jUnit tests fail on execution, the mutant is considered killed. In our experiments, we have thirteen mutation operators including all the default mutators (seven), three experimental mutators and three optional mutators [20, 31].

The machine we used for our experiments is a workstation with two Xeon E5- 2630V3 8 core CPUs @ 2.40GHz, 64GB DDR4 2133 MT/s memory, and a Samsung 850 EVO 500GB SSD.

III. EXPERIMENTAL RESULTS

In this section, we present our experimental results and discussion about our results. In Section III.A, we present the branch coverage results achieved by the original datasets. These results are considered to be the baseline results. In Section III.B, we present the branch coverage results achieved by the reduced datasets. These results are compared to the baseline results. In Section III.C, we present the mutation coverage results achieved by both of the original and reduced datasets.

A. Branch Coverage of the Original Datasets

Table III presents the branch coverage achieved by algorithms with original datasets. Among the datasets, *SupplyChain* consistently achieve higher coverage for all the algorithms. We observe that across algorithms, a considerable number of methods, and their branches were not executed, and thus the overall branch coverage appears to be considerably lower (<= 50%). This, however, can be explained as follows. Consider the branch coverage results of the OneR algorithm. The *SupplyChain* dataset achieves the highest branch coverage (57%), i.e., 51 out of 88 total branches. Among the missing 37 branches, 18 branches missed due to default configuration options. Seven branches are related to error handling, such as missing attribute values, and the remaining 12 branches cannot be covered as cross-validation is not performed while building models using the command-line interface (CLI).

To our surprise, *AustralianWeather* covers a significantly smaller number of branches (17) compared to the rest. This can be explained as follows: Among the five datasets, all the attributes of *AustralianWeather* belong to the nominal data type. All the attributes of *ForestCover, VideoGames, and Crime* belong to the numeric data type. In the case of *SupplyChain,* 3 out of 4 attributes belong to the numeric data type, and the remaining attribute belongs to the nominal data type. When executing the OneR algorithm with *AustralianWeather,* a method, newNumericRule(), was missed that has 36 branches and handles numeric attributes. Hence, *AustralianWeather* achieves a significantly lower branch coverage, whereas *SupplyChain* achieves the highest branch coverage, as it covers branches related to both numeric and nominal data types.

In our experiments, we executed the algorithms using WEKA's default configuration options only. This could cause branching conditions that are specific for other configuration options to be missed. As shown in [2], executing different configuration options could significantly increase branch coverage. Also, the branches related to error handling and GUI are not covered as we run our tests with clean datasets using the CLI.

We emphasize that, although branch coverage achieved by original datasets is not high, this does not affect the purpose of our experiments, which is to determine whether reduced datasets could achieve the same or similar coverage as the original dataset.

DATASETS	ALGORITHMS	# OF BRANCHES COVERED	TOTAL NUMBER OF BRANCHES	BRANCH COVERAGE
AustralianWeather	j48	180	750	24%
ForestCover		202		26%
SupplyChain		201		26%
VideoGames		202		26%
Crime		195		26%
AustralianWeather	Naïve Bayes	73	203	35%
ForestCover		77		37%
SupplyChain		99		48%
VideoGames		79		38%
Crime		78		38%
AustralianWeather	AdaBoost1	28	90	31%
ForestCover		17		18%
SupplyChain		28		31%
VideoGames		28		31%
Crime		28		31%
AustralianWeather	DecisionStump	50	128	39%
ForestCover		47		36%
SupplyChain		71		55%
VideoGames		48		37%
Crime		48		37%
AustralianWeather	OneR	17	88	19%
ForestCover		44		50%
SupplyChain		51		57%
VideoGames		45		51%
Crime		45		51%

TABLE III - BRANCH COVERAGE FOR ORIGINAL DATASETS

B. Branch Coverage of Reduced Datasets

In this section, we present the branch coverage results achieved by reduced datasets. For each dataset, we generate reduced datasets using two different approaches: random sampling and stratified sampling; we generate reduced datasets in four different sizes: 100 instances, 200 instances, 400 instances, and 800 instances, as discussed in Section II-D. Due to limited space, we present the median branch coverage achieved by each size relative to their baseline coverage.

Tables IV and V present the branch coverage results of reduced datasets generated using random sampling and stratified sampling, respectively. All the coverage results presented here are relative to their corresponding baseline. i.e., a relative branch coverage of 1.0 suggests that a reduced dataset achieves a branch coverage identical to the original dataset. Note that, in Tables IV and V, 39 out of 50 reduced datasets of size 800 produced by both random and stratified sampling, achieved branch coverages identical to the baseline; for the remainder of the cases, we notice the coverages do not significantly vary among different sample sizes. Therefore, in our experiments we did not consider sample size larger than 800 instances.

The results indicate that, for the j48 algorithm, reduced datasets of size 800 instances produced by both random and stratified sampling of *ForestCover, SupplyChain, and VideoGames* can retain their baseline branch coverage. For the NaiveBayes algorithm, the reduced versions of all five datasets can retain their branch coverage achieved by their respective original datasets and in some cases, reduced datasets achieving even higher branch coverage. Similarly, for the remaining three algorithms namely AdaBoost1, DecisionStump, and OneR, the reduced versions of all datasets except *Crime*, in most cases either retain their respective baseline branch coverage (1.0) or in some cases achieve a branch coverage closer to its baseline (0.9<=branch coverage<1.0).

For the reduced datasets of *Crime,* we observe that three out of five algorithms (j48, AdaBoost1, One-R) suffer from a loss in branch coverage. In particular, consider the case of j48 (Row 5 in Tables IV and V), which suffers from a significant loss in branch coverage. This is attributed to the class imbalance problem. The *Crime* dataset consists of 284,807 instances with two class labels: (0, 1); 99.82% instances belonging to Class 0 and remaining 0.18% belonging to Class 1. Due to class imbalance, chances of drawing all hundred samples (at random) that belong to Class 0 is higher.

In our experiments, for the reduced datasets of size 100 produced by random sampling, four out of five samples

have all their instances belonging to Class 0, and they achieve a relative median branch coverage of 0.12. On the contrary, three out of five reduced datasets of size 200 produced by random sampling have representation from both of the class labels, and they achieve a higher branch coverage comparatively (0.35). We notice that, in the case of j48, if a reduced dataset consists of a single label, there is a significant loss in branch coverage.

Next, we compare the coverage results of random sampling and stratified sampling. Our results indicate that, in most cases, the datasets reduced using both random and stratified sampling can achieve the same branch coverage.

Datasets	Algorithms	Size of the reduced dataset			
		100	200	400	800
AustralianWeather	j48	0.75	0.75	0.71	0.71
ForestCover		1.00	1.00	1.00	1.00
SupplyChain		0.81	0.73	0.92	1.00
VideoGames		0.96	0.96	0.96	1.00
Crime		0.12	0.35	0.12	0.35
AustralianWeather	Naïve Bayes	1.00	1.00	1.00	1.00
ForestCover		1.03	1.03	1.03	1.03
SupplyChain		1.00	1.00	1.00	1.00
VideoGames		1.00	1.00	1.00	1.00
Crime		1.00	1.00	1.00	1.00
AustralianWeather	AdaBoost1	1.00	1.00	1.00	1.00
ForestCover		1.78	1.67	1.00	1.00
SupplyChain		1.00	1.00	1.00	1.00
VideoGames		1.00	1.00	1.00	1.00
Crime		0.65	0.65	0.65	0.65
AustralianWeather	DecisionStump	0.95	0.95	0.95	0.95
ForestCover		1.00	1.00	1.00	1.00
SupplyChain		1.00	1.00	1.00	1.00
VideoGames		1.00	1.00	1.00	1.00
Crime		0.97	1.00	0.97	1.00
AustralianWeather	OneR	0.95	0.95	0.95	0.95
ForestCover		1.00	1.00	1.00	1.00
SupplyChain		0.96	0.96	0.96	0.96
VideoGames		1.00	1.00	1.00	1.00
Crime		0.76	0.92	0.76	1.00

Table IV – Relative Branch Coverage of reduced datasets (Random Sampling)

Datasets	Algorithms	Size of the reduced dataset			
		100	200	400	800
AustralianWeather	j48	0.75	0.75	0.71	0.71
ForestCover		1.00	1.00	1.00	1.00
SupplyChain		0.81	0.92	0.96	1.00
VideoGames		0.92	0.96	1.00	1.00
Crime		0.12	0.12	0.12	0.35
AustralianWeather	Naïve Bayes	1.00	1.00	1.00	1.00
ForestCover		1.03	1.03	1.03	1.03
SupplyChain		1.00	1.00	1.00	1.00
VideoGames		1.00	1.00	1.00	1.00
Crime		1.00	1.00	1.00	1.00
AustralianWeather	AdaBoost1	1.00	1.00	1.00	1.00
ForestCover		1.78	1.67	1.78	1.67
SupplyChain		1.00	1.00	1.00	1.00
VideoGames		1.00	1.00	1.00	1.00
Crime		0.65	0.65	0.65	1.00
AustralianWeather	DecisionStump	1.00	1.00	1.00	1.00
ForestCover		1.00	1.00	1.00	1.00
SupplyChain		1.00	1.00	1.00	1.00
VideoGames		1.00	1.00	1.00	1.00
Crime		0.97	0.97	0.97	1.00
AustralianWeather	OneR	0.95	0.95	0.95	0.95
ForestCover		1.00	1.00	1.00	1.00
SupplyChain		0.96	0.96	1.00	1.00
VideoGames		1.00	1.00	1.00	1.00
Crime		0.76	0.76	0.76	0.92

Table V – Relative Branch Coverage of Reduced Datasets (Stratified Sampling)

In the cases of *AustralianWeather*, *SupplyChain* and *VideoGames*, the datasets reduced using both random and stratified sampling achieves identical branch coverage. This can be explained by the fact that all reduced datasets have a good class label representation. For example, all five sample datasets of *AustralianWeather* of size 100 that are reduced using stratified sampling have the following class label distribution: 78 instances belong to *No*, and 22 instances belong to *Yes*. In the case of random sampling, amongst five samples, sample 5 consists of 86 instances belong to *No* and 14 instances belongs to *Yes* whereas, Sample 3 consists of 74 instances belong to *No* and 26 instances belongs to *Yes*.

Our results indicate that the reduced datasets of *ForestCover* generated using both random and stratified sampling achieve the same branch coverage as the original datasets across all algorithms. In comparison, the reduced datasets generated from *AustralianWeather*, *SupplyChain*, *VideoGames*, and *Crime* suffer from a minimal to moderate coverage loss in at least one of the five algorithms. This may be attributed to the fact, *ForestCover* is a multilabel dataset (7 class labels), whereas the rest of the four datasets are binary label dataset. More experimental data is required to obtain a better understanding. Also, our results indicate that in the case of the AdaBoost1 algorithm, the reduced datasets achieve a better branch coverage compared to the baseline, i.e., the original datasets. To some extent, this result is surprising, given the significant increase in branch coverage. This is possible because the reduced datasets may trigger execution scenarios that are different than the original datasets.

In the case of the *Crime* dataset, three algorithms suffer from a coverage loss. In particular, consider the coverage achieved by the reduced datasets of *Crime* produced by both random and stratified sampling. Row 5 in Tables IV and V indicates that the reduced dataset of size 200 produced by random sampling achieves a higher branch coverage (0.35) compared to the reduced dataset produced by stratified sampling of the same size (0.12). This can be attributed to the representativeness of the class label. On examination of reduced datasets, we observe that three out of five samples generated using random sampling have instances belonging to two class labels (Class 0 and Class 1). However, in the case of datasets reduced using stratified sampling, all instances belong to a single class (Class 0). Hence, subject programs achieve lower coverage while executing with stratified samples as they fail to trigger the execution of certain branches. The branch coverage results of the OneR algorithm suggest a similar pattern, i.e., the reduced dataset of size 200 produced by random sampling achieves a higher coverage (0.92) compared dataset reduced using stratified sampling of the same size (0.76).

This behavior of stratified sampling, i.e., all the instances of a reduced dataset belonging to a single class, is expected as it draws samples in a way that maintains the class distribution of the original dataset. Recall that the *Crime* dataset consists of 284,807 instances with two class labels: (0, 1); 99.82% instances belonging to Class 0 and remaining 0.18% belonging to Class 1. To generate a reduced dataset of size 200 instances using stratified sampling, instances are drawn in the following way (99.82% * 200) > 199 (instances) belonging to Class 0 and (0.18% * 200) < 1 (instances) belonging to Class 1. Hence, all the

instances belong to Class 0 and thus, the reduced dataset suffers from lack of class representativeness.

For the *Crime* dataset, a minimum of 556 instances is required to guarantee that a reduced dataset (stratified sampling) consists of instances belonging to both classes (0 and 1). Among four different sizes (100, 200, 400, and 800) of reduced datasets generated using stratified sampling, in three groups (100,200 and 400), all instances belong to class 0 and thus achieve a low branch coverage (0.12). In the case of reduced datasets of 800 instances, all five samples consist of instances of both classes and thus achieve a relatively higher branch coverage (0.35).

Our results indicate that approximately 80% of the reduced datasets achieve coverage identical or similar to the original datasets. In another word, the volume of a dataset does not directly attribute to branch coverage. Instead, factors such as lack of representativeness of class labels in a reduced dataset could impact branch coverage. The results suggest that in most cases, reduced datasets do not suffer from branch coverage loss. In this respect, they can be used in place of the original datasets to speed up the testing process.

Among the two sampling approaches, the results indicate that in most cases (around 75%) reduced datasets generated using both random and stratified sampling exhibit identical behavior. However, when a tester decides to use stratified sampling, he/she should choose the size of the reduced dataset (minimum number of samples) based on the original class distribution such that each class label is represented in the reduced dataset.

C. Mutation Coverage of Reduced Datasets

In this section, we present the mutation coverage results achieved by algorithms while executing with reduced datasets.

Given the size of the datasets and the number of mutants generated for SUT, the overall execution time can be between a few hours to several days. Due to time constraints, our experiments have an execution time limit of 48 hours (chosen arbitrarily). If a dataset takes more than 48 hours to complete, then we kill the test execution and use a relatively smaller dataset (10000 instances) as our baseline. Out of 20 baseline test executions, one baseline execution, j48 algorithm with the *VideoGames* dataset executed for more than 2 days. Hence, we generated five smaller samples of *VideoGames* dataset with 10000 instances each and used their median coverage as a baseline.

DATASETS	ALGORITHMS	SIZE OF THE REDUCED DATASET			
		100	200	400	800
AustralianWeather	j48	0.50	0.50	0.44	0.50
ForestCover		0.96	0.96	0.96	1.00
SupplyChain		0.64	0.57	0.71	0.79
VideoGames		0.88	0.88	0.92	0.96
Crime		0.14	0.24	0.14	0.24
AustralianWeather	Naïve Bayes	0.94	0.94	0.94	0.94
ForestCover		1.00	1.00	1.00	1.00
SupplyChain		1.00	1.00	1.00	1.00
VideoGames		1.00	1.00	1.00	1.00
Crime		1.00	1.00	1.00	1.00
AustralianWeather	AdaBoost1	1.00	1.00	1.00	1.00
ForestCover		1.92	1.38	1.00	1.00
SupplyChain		1.00	1.00	1.00	1.00
VideoGames		1.04	1.00	1.00	1.00
Crime		0.50	0.54	0.50	0.54
AustralianWeather	DecisionStump	1.00	1.00	1.00	1.00
ForestCover		1.03	1.00	1.00	1.00
SupplyChain		1.00	1.00	1.00	1.00
VideoGames		1.00	1.00	1.00	1.00
Crime		0.85	0.94	0.85	0.94
AustralianWeather	OneR	0.93	0.93	0.93	0.93
ForestCover		0.97	1.00	1.00	1.00
SupplyChain		1.07	1.07	1.07	1.07
VideoGames		0.97	0.97	1.00	1.00
Crime		0.66	0.77	0.66	0.89

TABLE VI - RELATIVE MUTATION COVERAGE OF REDUCED DATASETS (RANDOM SAMPLING)

DATASETS	ALGORITHMS	SIZE OF THE REDUCED DATASET			
		100	200	400	800
AustralianWeather	j48	0.50	0.50	0.50	0.50
ForestCover		0.92	0.96	1.00	0.96
SupplyChain		0.64	0.71	0.79	0.79
VideoGames		0.54	0.88	0.96	0.96
Crime		0.14	0.14	0.14	0.24
AustralianWeather	Naïve Bayes	0.94	0.94	0.94	0.94
ForestCover		1.00	1.00	1.00	1.00
SupplyChain		1.00	1.00	1.00	1.00
VideoGames		1.00	1.00	1.00	1.00
Crime		1.00	1.00	1.00	1.00
AustralianWeather	AdaBoost1	1.00	1.00	1.00	1.00
ForestCover		2.00	1.38	2.00	1.38
SupplyChain		1.00	1.00	1.00	1.00
VideoGames		1.00	1.00	1.00	1.00
Crime		0.50	0.50	0.50	1.00
AustralianWeather	DecisionStump	1.00	1.00	1.00	1.00
ForestCover		1.03	1.00	1.03	1.00
SupplyChain		1.00	1.00	1.00	1.00
VideoGames		1.00	1.00	1.00	1.00
Crime		0.85	0.85	0.85	0.97
AustralianWeather	OneR	0.93	0.93	0.93	0.93
ForestCover		1.00	1.00	1.00	1.00
SupplyChain		1.07	1.07	1.13	1.07
VideoGames		0.97	1.00	0.97	1.00
Crime		0.66	0.66	0.66	0.77

TABLE VII-RELATIVE MUTATION COVERAGE OF REDUCED DATASETS (STRATIFIED SAMPLING)

Tables VI and VII present the mutation coverage results of the reduced datasets. All the coverage results presented here are relative to their corresponding baseline. The results from Tables VI and VII suggest that the j48 algorithm performs poorly with the reduced datasets of *AustralianWeather* and *SupplyChain.* Similarly, the reduced datasets of *Crime* result in a mutation coverage decrease for all the algorithms except Naive Bayes. The rest of the reduced datasets generated using both random and stratified sampling can retain their baseline mutation coverage.

We report that the majority of the mutation coverage results (except reduced datasets of *SupplyChain* on j48) mirrors with their respective branch coverage results (Table IV & V; Table VI & VII). Figures 1 and 2 present a correlation graph of branch coverage vs. mutation coverage for random sampling and stratified sampling, respectively. In Figures 1 and 2, x-axis indicates branch coverage, and the y-axis indicates mutation coverage. For the datasets reduced via random sampling, branch vs. mutation coverage has a Pearson correlation coefficient of 0.944148, whereas the datasets reduced via stratified sampling has a fractionally lower Pearson correlation coefficient of 0.939506. The result suggests that in most cases, mutation coverage has a strong positive correlation with the branch coverage. To our surprise, the mutation results of j48 using the *SupplyChain* dataset reduced using stratified sampling does not appear to

correlate well with branch coverage, and we plan to investigate this further as part of our future work.

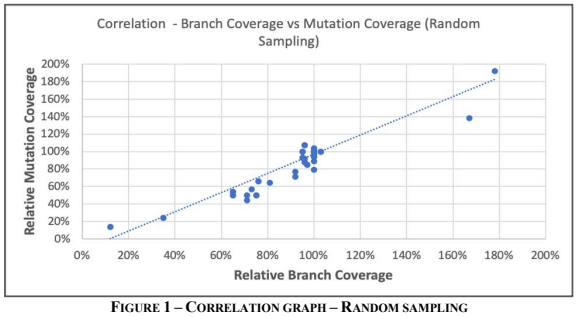

FIGURE 1 – CORRELATION GRAPH – RANDOM SAMPLING

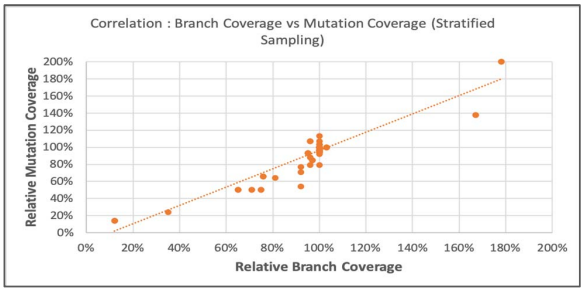

FIGURE 2 – CORRELATION GRAPH – STRATIFIED SAMPLING

IV. THREATS TO VALIDITY

Threats to internal validity are factors that may be responsible for experimental results, without our knowledge. To reduce human errors in the experimental procedure, we tried to automate our experiments as much as possible. In particular, we wrote scripts to automatically execute tests, measure code and mutation coverage, and generate coverage reports. Further, the results generated from samples of each dataset were verified manually, whenever possible.

Threats to external validity occur when the experimental results could not be generalized to other subjects. Using a single dataset for our experiments might impact the validity of our results due to lack of representativeness. To mitigate this threat, we used four supervised learning algorithms from WEKA that belong to different groups and five datasets from different application domains. More experiments using other learning algorithms, including both supervised and unsupervised algorithms, and other datasets, can further reduce the threats to external validity.

V. RELATED WORK

First, we review existing work reported on testing machine learning algorithms. One challenge in testing machine learning algorithms is how to deal with the test oracle problem. Murphy et al. [4,5] proposed a metamorphic testing technique to test machine learning algorithms. They developed metamorphic properties for three machine learning algorithms, including MartiRank, SVMLight, and PAYL. Similarly, Nakajima et al. [7] proposed a systematic approach to derive metamorphic properties and translation functions for testing a special class of classifiers known as Support Vector Machines (SVM). Xie et al. [11] proposed a metamorphic testing approach to test supervised learning algorithms, namely Naïve Bayes classifier and k-nearest neighbor classifier. Our work differs from these works in that we focus on evaluating the effectiveness of using smaller datasets in testing supervised learning algorithms.

Next, we review existing work on dataset reduction for big data applications [3, 6, 8, 13, 14, 15, 16, 17]. Such work is relevant because many machine learning algorithms are big data applications in that they are designed to learn from large amounts of data. Ur Rehman et al. [13] reviewed existing data reduction techniques such as compression-based data reduction method, dimension reduction techniques for big data applications. Czarnowski et al. [14] proposed an agent-based population learning algorithm for data reduction. Their algorithm aims at finding a subset of the original dataset that can be used to build a classifier that is similar to the classifier built using the original dataset. In contrast, our work focuses on volume reduction and its impact on test effectiveness.

Rojas et al. [38] investigate how different sampling strategies could impact data exploration on big datasets by comparing the performance of smaller datasets generated using random sampling and three non-random sampling techniques namely Query by committee, Density, and Uncertainty sampling. These works try to discover the same amount of information with a reduced dataset, which is different from our work, which tries to find a subset of the original dataset that preserves test effectiveness. To the best of our knowledge, our work is the first to investigate the effectiveness of dataset reduction in testing machine learning algorithms.

Finally, we mention that there are studies in the literature that investigate the effect of test suite minimization on fault detection effectiveness [33, 34, 35, 36, 37]. A test suite is different than a dataset, as the former is a set of test cases each of which represents an independent test input, whereas the latter is a set of instances that are together used as one single test input.

VI. CONCLUSION AND FUTURE WORK

In this paper, we report a study that investigates the use of reduced datasets in testing machine learning algorithms. We used four supervised learning algorithms from WEKA as our subject programs. Five publicly available datasets from Kaggle.com were chosen as subject datasets. For each dataset, we generated reduced datasets in four different sizes using random and stratified sampling. Then, we executed the algorithms with the original and the reduced datasets and measured test effectiveness in terms of branch and mutation coverage. Our results indicate, in most cases, reduced datasets of very small sizes (e.g. 800 instances) can retain branch and mutation coverage of the original, big datasets (e.g., >100,000 instances). This suggests that reduced datasets can be used to effectively test machine learning algorithms. Our results also indicate a high correlation between branch coverage and mutation coverage. Thus, branch coverage can be used when mutation testing is prohibitively expensive.

This is the first step in our larger effort to speed up testing machine learning algorithms. We plan to continue our work in the following directions. First, we plan to investigate the reduction of even bigger multi-label datasets (> 1 GB) and its effect on testing machine learning algorithms. Second, we plan to expand our study to include unsupervised learning algorithms. Compared to supervised

learning algorithms, unsupervised learning algorithms learn from unlabeled datasets and thus could be harder to validate its output. Third, our experiments show that there exists a high correlation between branch and mutation coverage. However, some recent work reports that traditional code coverage measures such as branch coverage may not be adequate for testing deep learning algorithms. We believe that this has to do with the nature of the algorithms and also the types of fault that may exist in the algorithms. We plan to study this further by conducting experiments on deep learning algorithms. Finally, we plan to develop new methods, i.e., methods other than random sampling, for dataset reduction. For example, how to perform equivalence partitioning among instances in a big dataset, and then choose one or more representatives from each equivalence group.

VII. Acknowledgement

This work is supported by research grant (70NANB18H207) from Information Technology Lab of National Institute of Standards and Technology (NIST).

Disclaimer: Certain software products are identified in this document. Such identification does not imply recommendation by the NIST, nor does it imply that the products identified are necessarily the best available for the purpose.

References

[1] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten (2009). The WEKA Data Mining Software: An Update. SIGKDD Explorations, Volume 11, Issue 1.

[2] Chandrasekaran, Jaganmohan, et al. "Applying combinatorial testing to data mining algorithms." *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2017.

[3] Feldman, Dan, Melanie Schmidt, and Christian Sohler. "Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering." *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2013.

[4] Murphy, Christian, Gail E. Kaiser, and Marta Arias. "An approach to software testing of machine learning applications." (2007).

[5] Murphy, Christian, Gail E. Kaiser, and Lifeng Hu. "Properties of machine learning applications for use in metamorphic testing." (2008).

[6] Kira, Kenji, and Larry A. Rendell. "The feature selection problem: Traditional methods and a new algorithm." *Aaai*. Vol. 2. 1992.

[7] Nakajima, Shin, and Hai Ngoc Bui. "Dataset coverage for testing machine learning computer programs." *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2016.

[8] Khalid, Samina, Tehmina Khalil, and Shamila Nasreen. "A survey of feature selection and feature extraction techniques in machine learning." *2014 Science and Information Conference*. IEEE, 2014.

[9] Datasets Documentation, https://www.kaggle.com/docs/datasets.

[10] Hotness calculation formula, https://www.kaggle.com/general/39290

[11] Xie, Xiaoyuan, et al. "Testing and validating machine learning classifiers by metamorphic testing." *Journal of Systems and Software* 84.4 (2011): 544-558.

[12] Zhang, Zhiyi, and Xiaoyuan Xie. "Towards testing big data analytics software: the essential role of metamorphic testing." *Biophysical reviews* 11.1 (2019): 123-125.

[13] ur Rehman, Muhammad Habib, et al. "Big data reduction methods: a survey." *Data Science and Engineering* 1.4 (2016): 265-284.

[14] Czarnowski, Ireneusz, and Piotr Jędrzejowicz. "An Approach to Data Reduction for Learning from Big Datasets: Integrating Stacking, Rotation, and Agent Population Learning Techniques." *Complexity* 2018 (2018).

[15] Czarnowski, Ireneusz, and Piotr Jędrzejowicz. "Stacking and rotation-based technique for machine learning classification with data reduction." *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*. IEEE, 2017.

[16] Liu, Qingzhong, et al. "Mining the big data: The critical feature dimension problem." *2014 IIAI 3rd International Conference on Advanced Applied Informatics*. IEEE, 2014.

[17] Wold. Svante, Kim Esbensen, and Paul Geladi. "Principal component analysis." *Chemometrics and intelligent laboratory systems* 2.1-3 (1987): 37-52.

[18] M. Hoffmann, B. Janiczak, E. Mandrikov and M. Friedenhagen. Jacoco code coverage tool. Online , 2016

[19] H. Coles. Pit mutation testing. http: //pitest.org/, 2016.

[20] Coles, Henry, et al. "Pit: a practical mutation testing tool for java." *Proceedings of the 25th International Symposium on Software Testing and Analysis*. ACM, 2016.

[21] LEGO Database, https://www.kaggle.com/rtatman/lego-database#inventory_parts.csv

[22] League of Legends Ranked Matches, https://www.kaggle.com/paololol/league-of-legends-ranked-matches#stats1.csv

[23] Rain in Australia, https://www.kaggle.com/jsphyg/weather-dataset-rattle-package

[24] As Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science

[25] Forest Cover Type Prediction, https://www.kaggle.com/c/forest-cover-type-prediction/overview

[26] Credit Card Fraud Detection, https://www.kaggle.com/mlg-ulb/creditcardfraud

[27] John, George H., and Pat Langley. "Estimating continuous distributions in Bayesian classifiers." *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1995.

[28] Freund, Yoav, and Robert E. Schapire. "Experiments with a new boosting algorithm." *icml*. Vol. 96. 1996.

[29] Holte, Robert C. "Very simple classification rules perform well on most commonly used datasets." *Machine learning* 11.1 (1993): 63-90.

[30] Salzberg, Steven L. "C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993." *Machine Learning* 16.3 (1994): 235-240.

[31] Mutation Operators, https://pitest.org/quickstart/mutators/

[32] OneDrive, https://1drv.ms/f/s!AjZ3W-Mz9wPKhtlLoWUU2zZKzm4bRg

[33] Wong, W. Eric, et al. "Effect of test set minimization on fault detection effectiveness." *Software: Practice and Experience*28.4 (1998): 347-369.

[34] Wong, W. Eric, et al. "Test set size minimization and fault detection effectiveness: A case study in a space application." *Proceedings Twenty-First Annual International Computer Software and Applications Conference (COMPSAC'97)*. IEEE, 1997.

[35] Rothermel, Gregg, et al. "An empirical study of the effects of minimization on the fault detection capabilities of test suites." *Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)*. IEEE, 1998.

[36] Jones, James A., and Mary Jean Harrold. "Test-suite reduction and prioritization for modified condition/decision coverage." *IEEE Transactions on software Engineering* 29.3 (2003): 195-209.

[37] Rothermel, Gregg, et al. "Empirical studies of test-suite reduction." *Software Testing, Verification and Reliability* 12.4 (2002): 219-249.

[38] Rojas, Julian A., Ramos, et al. "Sampling techniques to improve big data exploration." *2017 IEEE 7th symposium on large data analysis and visualization (LDAV)*. IEEE, 2017.