Evaluation of the existing Metamorphic Relations for Machine Learning Classifiers from a Whitebox Coverage perspective

Sadia Ashraf[*], Dr. Aamir Nadeem[**], Dr. Salma Imtiaz[**]

[*] Department of Software Engineering, IIUI
[**] Department of Software Engineering, CUST

***Abstract*-** Machine learning (ML) is one of the most popular area in the field of AI, which has led to a rapid rise in ML applications. Most of these applications use libraries like Sci-kit learn and Weka which implement these algorithms. Testing these ML applications is difficult because they suffer from the oracle problem and traditional testing techniques generally do not work without oracles. Metamorphic testing is used instead of the traditional testing techniques for these applications. Although the code for the actual application is usually too simple to be need formal testing but the implementation of the algorithms within the libraries is not. This work evaluates the existing metamorphic relations in the literature for their effectives in detecting faults and finds the kill rate of these metamorphic relations when they are used to test the implementations for 5 classifiers (ANN, ID3, KNN, Naïve Bayes, SVM) from Sci-kit learn. It also finds the statement and branch coverage when using these Metamorphic relations and finds an indirect relationship between coverage and kill rate which can be exploited to improve the kill rate for the test suites.

***Index Terms*-** Metamorphic Testing, Metamorphic Relations, Oracle, Machine Learning.

I. INTRODUCTION

Metamorphic Testing (MR) is a technique used for software which cannot be tested with conventional testing methods. It tests software based on its expected behavior as opposed to expected outputs, which makes it a popular choice for complex systems that do not have a fixed output for a given input and are characterized by randomness. MRs are used to test Autonomous Vehicles [1-9], where the possible noise in the environment is simulated and fed to the systems to test it for the issues it might face on the road. It is popular in testing stochastic software [10-15], that does not have oracles to guide the testing process. NLP based systems [9, 16-28] are tested with metamorphic tests due to the semantic nature of these systems. A fairly new area where metamorphic testing is being used is fairness testing [28-35], where the systems are tested for bias and discrimination. Most of these systems suffer from the oracle problem. The oracle problem makes it difficult to test the systems because the correct outputs are not known in advance to be able to compare the outcomes of the test cases with these outcomes. Testing without oracles might result in insufficient testing since the traditional coverage-based techniques are replaced by metamorphic relations.[36, 37]

Metamorphic Relations are used in the absence of oracles where normal testing techniques like Blackbox and White box testing cannot be applied. Metamorphic Testing is used to derive expected outputs, when the input and the expected behavior is known. The need for oracles is eliminated, when MRs are used to test the system. The accuracy of the system is determined by its conformance to the MR and if the system deviates from the expected behavior then it indicates a fault in the system. Metamorphic Testing is useful for regression testing [38, 39], where the system is tested for changes in the output after the updates are made in it. Machine Learning Systems are generally tested with MRs, since these systems do not have oracles to guide the testing process.

Machine Learning systems like classifiers are used to make predictions based on the patterns learned from the given data. The expected output for them is no available and therefore these systems are tested with MRs'. The popular MRs' for classifiers are the MRs by Xie that are based on the necessary properties of ML Classifiers. Xie's [13] MRs are divided into six categories, each representing a property of the ML Classifiers. **Data Transformation**, applies affine transformation on the data. **Data Permutation,** applies permutations to the classes, **Attribute Addition,** adds uninformative attributes to the dataset. **Data and Prediction augmentation** repeats the rows in such a way as to not affect prediction. **Class Addition and Modification** duplicates data in such a way that the accuracy is not affected. **Class Removal** deletes records, etc so the accuracy remains unaffected.

These MRs work by creating a model by training the algorithm with the original dataset. The follow-up dataset is created by modifying it according to the MRs and the follow-up model is created with the modified dataset. Both the models should work exactly in the same way because the modifications introduced by Xie's MRs[13] do not affect the correlations between the attributes. Therefor any variation in the predictions made by the two models is due to the faults in the system. The faults are located when the follow-up and original models vary in accuracy.

The performance of the Test Suite in detecting faults is known as the fault detection effectiveness. The higher the effectiveness of a Test Suite the better it is at detecting faults. Fault detection effectiveness is measured in literature [37, 40, 41] using mutation testing. Mutation testing adds artificial faults called mutants to the code and then runs the test suite on it to see if the fault is detected. The number of mutants that the Test Suite detects, determines the kill rate of the test suite.

This work evaluates the effectiveness of Xie's MRs by applying mutation testing on the MRs. It also measures **Statement** and **Branch** coverage of the code when the MRs are run. The code used for this work is the implementation of the sci-kit learn's classifiers. Five most commonly used classifiers [42-46] that are used for this study are Naïve Bayes, ID3 (Iterative Dichotomiser), SVM (Support Vector Machine), KNN (K-Nearest Neighbor) & ANN (Artificial Neural Network).

Xie's MRs consists of 11 MRs which cover different aspects of the machine learning algorithms. We evaluated all 11 MRs with 5 different algorithms and two different datasets. This work reports the kill rate for the MRs and the possible cause for the kill rate being low from a white box coverage point of view. The motivation for this work is to find the correlation between the kill rate and the uncovered mutants. The findings from this work can be used explore the direction of coverage for machine learning algorithms. Future work to improve the effectiveness of the MRs can be done based on the findings in this work.

The observations made in our work are that most mutants are never executed because they are in the uncovered part of the code. The MR does not exercise almost 50% of the code for the algorithm. The kill rate for the MRs ranges from 0% to 21% depending on the number of mutants that are covered and the number of mutants that are added to the code.

Generally, the code coverage does not have a direct correlation with the mutant kill rate [37]. The kill rate depends on multiple factors like the type and diversity of the mutants used along with the quality of the test suite. But the number of mutants being killed is directly dependent on the no of mutants that are exercised. If the mutants are never exercised then they do not contribute to the kill rate of the Test Suite. Any MRs that can improve the coverage might result in a test suite with a higher kill rate and is more effective in detecting faults in general.

The rest of this paper contains the background where existing work related to this study is discussed. After that the actual study is discussed along with the observations made in the accuracy, kill rate and coverage section. The conclusion and future work sections discuss the possible issues with this work and the direction the future work can take.
article guides a stepwise walkthrough by Experts for writing a successful journal or a research paper starting from inception of ideas till their publications. Research papers. are highly recognized in scholar fraternity and form a core part of PhD curriculum. Research scholars publish their research work in leading journals to complete their grades. In addition, the published research work also provides a big weight-age to get admissions in reputed varsity. Now, here we enlist the proven steps to publish the research paper in a journal.

II. BACKGROUND

This section explains basics of Machine learning, Metamorphic testing along with the formal definitions for the techniques being used in this work. Related work in the area is also discussed.

Definitions:

Metamorphic Testing

In a system where f is the target algorithm or function. The metamorphic relation (MR) can be described as a property of f over a sequence of two or more inputs, denoted as $(a_1, a_2, ..., ai)$, for $i > =2$, with the corresponding outputs for the sequence of inputs are $(f(a_1), f(a_2), ..., f(ai))$.

Metamorphic relations are a relation $R \subseteq X_1 \times X_2 \times ... \times Xi \times Y_1 \times Y_2 \times ... \times Yi$, where $\subseteq$ consists of a subset of relations. $X_1 \times X_2 \times ... \times Xi$ *and* $Y_1 \times Y_2 \times ... \times Yi$ is the Cartesian product of the input domain and the output domain. Therefore, we can say $R(a_1, a_2, ..., ai, f(a_1), f(a_2), ..., f(a_i))$ to indicate that the tuple $(a_1, a_2, ..., ai, f(a_1), f(a_2), ..., f(a_i))$ belongs to R.

Machine Learning Classifiers

Formally, a classifier is a mapping of a function $\boldsymbol{C: X \rightarrow Y}$, where X is a set of the input domain or feature space, and $\boldsymbol{Y}$ is the output space, which contains the class labels that the model can assign to the given input. For a given input vector $\boldsymbol{x \in X}$, the classifier assigns a class label $\boldsymbol{y \in Y}$ to that instance, hence making a prediction based on the data it was trained on. The formal definitions for the 5 classifiers used in this work are given below:

KNN (k-Nearest Neighbors):

If $X = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ is a training dataset where xi is the vector of feature and y_i is the output label for that feature. Where $x_i \in \mathbb{R}^n$ and $y_i \in Y$, which is a set of all the possible class labels. Given a distance metric d(x, x'), where d: $\mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ represents the distance between two feature vectors, the KNN algorithm calculates the distances between the new instance x^* and all training instances: $D = \{(d(x^*, x_1), y_1), (d(x^*, x_2), y_2), ..., (d(x^*, x_n), y_n)\}$, where D is the set of Euclidean distances sorted in ascending order and the Euclidean distance between vectors x and x' is

$$d(x, x') = \sqrt{(\sum(x_i - x'_i)^2)}$$

ID3 (Iterative Dichotomiser 3):

For a given dataset $X = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where $x_i \in \mathbb{R}^n$ consists of the features or attributes of X and $y_i \in Y$ is a set of all possible class label for X. ID3 recursively partitions the feature space based on information gain, which in turn depends

on the entropy of the chaos in the given data. The Entropy of a target variable is given below:

$$Y: H(Y) = -\sum(p(y) * log_2(p(y)))$$

The Information gain for a feature X, given the target Y is given by the formula below:

$$IG(X, Y) = H(Y) - \sum((|X_v| / |X|) * H(Y|X_v))$$

where $|X_v|$ is the number of instances in X with value v.

ANN (Artificial Neural Networks):

For a given dataset $X = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where $x_i \in \mathbb{R}^n$ is a set or real number that represent the feature vector for X and $y_i \in Y$ is the corresponding target class label. Given an input vector x, the forward propagation in ANN follows the steps below:

The input layer is initialized with the value of x. Each layer in the hidden layers and the output layer's activation formula is computed using the formulae below:

For each hidden layer neuron j:

$$z_j = \sum(w_{ji} * a_i) + b_j \; a_j = \sigma(z_j)$$

For each output layer neuron k:

$$z_k = \sum(w_{kj} * a_j) + b_k \; a_k = \sigma(z_k)$$

where w_{ji} and w_{kj} are the weights, b_j and b_k are the biases, a_i and a_j are the activations of the input and hidden layer neurons, respectively, and $\sigma(\cdot)$ is the activation function.

SVM (Support Vector Machine):

For a given dataset $X = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where $x_i \in \mathbb{R}^n$ is a vector made of real numbers that make up the features of the dataset and $y_i \in Y$ is the set of target class label. SVM finds an optimal hyperplane in the feature space in order to separate the data points from various different classes with a maximum margin. For X, the SVM is defined as follows:

For x_i, the feature vector is mapped into a higher-dimensional feature space with the following kernel function, where $\mathcal{H}$ is a high-dimensional space.

$$\phi: \mathbb{R}^n \rightarrow \mathcal{H}$$

$\mathcal{H}$ contains the optimal hyperplane which maximizes the margin between the support vectors, while the SVM problem is formulated as follows:

$$min \; \tfrac{1}{2}||w||^2 + C\sum\xi_i \; s.t. \; y_i(w \cdot \phi(x_i) + b) \geq 1 - \xi_i \; \xi_i \geq 0$$

where w is the weight vector, b is the bias term, C is the regularization parameter, ξ_i are the slack variables, and y_i represents the target class label (+1 or -1).

New instances are classified based on the sign of

$$(w \cdot \phi(x) + b).$$

Naive Bayes:

For a given dataset $X = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y,$ where $x_i = (x_{i1}, x_{i2}, ..., x_{im}) \in \mathbb{R}^m$ is a set of real numbers that make up the feature vectors of the datset and $y_i \in Y$, where Y is the set of target class labels. For P(y) being the prior probabilities for each class y based on X, the conditional probabilities are $P(x_j|y)$ for each feature x_j given each class y. New instances x^* are classified by calculating the posterior probabilities $P(y|x^*)$ for each class y and selecting the class with the highest probability using the given formulae

Bayes' theorem:

$$P(y|x) = (P(x|y) * P(y)) / P(x)$$

Naive Bayes assumption (feature independence):

$$P(x|y) = \prod P(x_j|y)$$

Xie's Metamorphic Relations

Xie et al presented 11 metamorphic relations for machine learning classifiers. These metamorphic relations represent the necessary properties of Machine learning algorithms. The following table discusses the MRs in detail.

Xie's Metamorphic Relations			
Sr no	**Name**	**Category**	**Description**
1	Consistence with affine transformation	Transformation	This MR states that the ML model should be immune to any affine transformations that are applied to the dataset. If we multiply any column with a constant number and add a constant number to it. The model trained on the new dataset should be no different from the original model.
2	Permutation of class labels	Permutation	If the class labels in the dataset are swapped in a 'one to one' manner than the new dataset should give the same output as the original dataset.

3	Permutation of the attribute		If the values of the attributes are shuffled within the training set as well as the test set, then the model created with the follow-up dataset should behave no different than the original model.
4	Addition of uninformative attributes	Attribute Addition	If a new attribute is added to the dataset that has no association or correlation with any class label, then this shouldn't affect the outcome of the follow-up model.
5	Addition of informative attributes		If an attribute is added to the training dataset that is strongly associated with only one class label and it is not associated with the other class labels then this should not affect the predictions made by the follow-up dataset.
6	Consistence with re-prediction	Prediction Augmentation	If we take a single test case along with its class label and append it to the training data. Then this will not affect the predictions for the same test case
7	Additional training sample		If a few test cases with a certain label along with their class labels are duplicated and added back to the dataset. This should not affect the output for the follow-up models predictions for that class.
8	Addition of classes by duplicating samples	Duplication	If all the samples which are associated with class labels accept one particular class label 'l' are duplicated and then their class labels are changed by appending a * with them then the unduplicated samples will still have the same prediction as before.
9	Addition of classes by re-labeling samples		If all the samples accept for the ones that are associated with a certain class label are relabeled, the predictions for the specific class should still remain the same.
10	Removal of classes	Class Removal	If an entire class is removed from the training data which is other than a specific class label 'l', then all the predictions made by the algorithm regarding 'l' should remain unaffected.
11	Removal of samples		If some random samples are removed from the training set along with their class labels where they are not associated with the class label 'l' then the model's predictions for 'l' should remain unchanged.

III. Accuracy, Coverage and Kill Rate

Accuracy for a model is the no of correct predictions made by a model as compared to the actual predictions [13, 47-50]. It is commonly used as a metric to evaluate machine learning classifiers [13, 47]. Xie's metamorphic relations are based on the consistency of the prediction when applying the MRs. Therefore, if the MRs are applied correctly, the overall accuracy of the results should not change too much. The tables below give the accuracy for the classifiers when the MRs are applied to them with two datasets. There is very little change in the accuracy of the test suites when the MRs are applied.

ACCURACIES FOR THE CLASSIFIERS AFTER APPLYING METAMORPHIC RELATIONS WITH IRIS					
	SVM	**KNN**	**ANN**	**NB**	**ID3**
No MR	1.0	1.0	1.0	1.0	1.0
MR-0	0.93	.98	0.96	.96	0.98
MR-1.1	1.0	1.0	1	1	1

MR-1.2	1.0	.98	1	1	1
MR-2.1	0.98	1.0	1	1	1
MR-2.2	1.0	.98	0.98	.98	1
MR-3.1	1.0	1.0	.97	.98	1
MR-3.2	1.0	.98	.96	.98	1
MR-4.1	1.0	1.0	1	.97	1
MR-4.2	0.87	1.0	.96	.96	.98
MR-5.1	1.0	.98	1	1	1
MR-5.2	0.98	.97	.97	.97	.97

***Table 1**: Accuracy for Iris*

Since the accuracies do not change too much after the MRs' are applied, that shows that the Xie's MRs were correctly implemented and they are working as they should. If the accuracies change too much after the MRs are applied, that indicates there are issues, either in the implementation or the MR itself is faulty. The variation in the accuracy depends on the accuracy of the model as well.

ACCURACIES FOR THE CLASSIFIERS AFTER APPLYING METAMORPHIC RELATIONS WITH TITANIC					
	SVM	**KNN**	**ANN**	**NB**	**ID3**
No MR	1.0	1.0	1.0	1.0	1.0
MR-0	1.0	1.0	1.0	1.0	1.0
MR-1.1	1.0	1.0	1.0	1.0	1.0
MR-1.2	1.0	1.0	1.0	1.0	1.0
MR-2.1	1.0	1.0	1.0	1.0	1.0
MR-2.2	1.0	1.0	1.0	1.0	1.0
MR-3.1	1.0	1.0	1.0	1.0	1.0
MR-3.2	1.0	1.0	1.0	1.0	1.0
MR-4.1	1.0	1.0	1.0	1.0	1.0
MR-4.2	1.0	1.0	0.98	1.0	1.0
MR-5.1	1.0	0.98	1.0	0.98	0.98
MR-5.2	0.98	1.0	1.0	1.0	1.0

***Table 2:** Accuracy for Titanic*

Coverage is a metric often used in classic testing techniques [51-55] that is used to measure the extent to which the test cases cover the code. It represents the thoroughness of the testing process and provides the assurance that all or most parts of the system under test is covered at least once. It also is a good guideline to assess the need of additional testing. When metamorphic testing is used, coverage is not taken into consideration, which means that there is a possibility that a system needed more testing where the testing effort is stopped prematurely, or vice versa. The tables below show the coverage achieved when Xie's MRs are applied to test the code for the implementation of the chosen classifiers from the Scikit-learn library for python [42-46, 54]. The coverage values recorded here are for statement coverage and branch coverage, which are two types of white box coverage [40, 41, 56] i.e. the type of testing where the code of the system under test is available.

Statement coverage is a testing criterion that aims to cover every statement in the code at least once. It measures the lines of code that were exercised at least once out of the total number of lines in the code, whereas the branch coverage attempts to cover all the branches within the code at least once.

STATEMENT COVERAGE FOR THE CLASSIFIERS AFTER APPLYING METAMORPHIC RELATIONS WITH IRIS					
	SVM	**KNN**	**ANN**	**NB**	**ID3**
No MR	67/120 =55.8	59/37 = 62.7	156/362 = 43	63/123 = 51.22	124/265 = 46.7
MR-0	55.8	62.7	43	51.22	46.7
MR-1.1	55.8	62.7	43	51.22	46.7
MR-1.2	55.8	62.7	43	51.22	46.7
MR-2.1	55.8	62.7	43	51.22	46.7
MR-2.2	55.8	62.7	43	51.22	46.7
MR-3.1	55	62.7	43	51.22	46.7

MR-3.2	55.8	62.7	43	51.22	46.7
MR-4.1	55.8	62.7	43	51.22	46.7
MR-4.2	55.8	62.7	43	51.22	46.7
MR-5.1	55.8	62.7	43	51.22	46.7
MR-5.2	55.8	62.7	43	51.22	46.7

Table 3: *Statement Coverage for Iris*

It is important to note that the coverage, be it statement coverage or branch coverage is not correlated with the metamorphic relations in any way and when dealing with simpler code like that of KNN, achieving complete coverage can be a simple matter of adding a few more tests to the system. But when dealing with code that is complex and has a lot of nested condition within each other like the code for ID3 and ANN manually adding MTs becomes a tedious task and may cost a lot of time and energy to do so. The coverage mentioned here is the coverage that was recorded after training the sci-kit learn algorithms with the two datasets and then finding their accuracy.

STATEMENT COVERAGE FOR THE CLASSIFIERS AFTER APPLYING METAMORPHIC RELATIONS WITH TITANIC					
	SVM	**KNN**	**ANN**	**NB**	**ID3**
No MR	55.8	51.78	42.3	46.39	48.4
MR-0	55.8	51.78	42.3	46.39	48.4
MR-1.1	55.8	51.78	42.3	46.39	48.4
MR-1.2	55.8	51.78	42.3	46.39	48.4
MR-2.1	55.8	51.78	42.3	46.39	48.4
MR-2.2	55.8	51.78	42.3	46.39	48.4
MR-3.1	55.8	51.78	42.3	46.39	48.4
MR-3.2	55.8	51.78	42.3	46.39	48.4
MR-4.1	55.8	51.78	42.3	46.39	48.4
MR-4.2	55.8	51.78	42.3	46.39	48.4
MR-5.1	55.8	51.78	42.3	46.39	48.4
MR-5.2	55.8	51.78	42.3	46.39	48.4

Table 4: *Statement Coverage for Titanic*

The tables below show the branch coverage of the classifiers with Iris and Titanic. These tables show that the coverage achieved is not affected by dataset being used and since all the MRs make changes to the dataset, the coverage does not change much from one MR to another.

BRANCH COVERAGE FOR THE CLASSIFIERS AFTER APPLYING METAMORPHIC RELATIONS WITH IRIS					
	SVM	**KNN**	**ANN**	**NB**	**ID3**
No MR	0	16.66	5	31.2	5
MR-0	0	16.66	5	31.2	5
MR-1.1	0	16.66	5	31.2	5
MR-1.2	0	16.66	5	31.2	5
MR-2.1	0	16.66	5	31.2	5
MR-2.2	0	16.66	5	31.2	5
MR-3.1	0	16.66	5	31.2	5
MR-3.2	0	16.66	5	31.2	5
MR-4.1	0	16.66	5	31.2	5
MR-4.2	0	16.66	5	31.2	5
MR-5.1	0	16.66	5	31.2	5
MR-5.2	0	16.66	5	31.2	5

Table 5: *Branch Coverage for Iris*

Kill rate or the mutant kill rate is the measure of the effectiveness of the test suite. It is basically a testing technique to evaluate test suite. It is a metric that represents the ability of a test suite to uncover faults or mutants. Mutants are generally artificial faults fed into the system by making small changes in the code that do not result in syntax errors but introduce faults into the system.

BRANCH COVERAGE FOR THE CLASSIFIERS AFTER APPLYING METAMORPHIC RELATIONS WITH TITANIC					
	SVM	KNN	ANN	NB	ID3
No MR	0	16.66	5.2	31.25	6.25
MR-0	0	16.66	5.2	31.25	6.25
MR-1.1	0	16.66	5.2	31.25	6.25
MR-1.2	0	16.66	5.2	31.25	6.25
MR-2.1	0	16.66	5.2	31.25	6.25
MR-2.2	0	16.66	5.2	31.25	6.25
MR-3.1	0	16.66	5.2	31.25	6.25
MR-3.2	0	16.66	5.2	31.25	6.25
MR-4.1	0	16.66	5.2	31.25	6.25
MR-4.2	0	16.66	5.2	31.25	6.25
MR-5.1	0	16.66	5.2	31.25	6.25
MR-5.2	0	16.66	5.2	31.25	6.25

Table 6: *Branch Coverage for Titanic*

The better a test suite is the higher its kill rate will be. The tables below show the kill rate for Xie's MRs for the the five classifiers used in the study. The mutants are added in such a way that they are evenly distributed throughout the code. Although it is not possible to add mutants to the parts of the code that use built in function calls and do not have many mutant operators that can be modified. This is what happened with SVM and KNN. The green parts of the tables represent the mutants that are killed by the MRs and the red part represents the mutants that are in the part of the code that is not covered by either statement or branch coverage.

Kill rate for the metamorphic relations for ANN with the Titanic Dataset Total lines of Code: 1518												
Sr no	**Mutant**	**MR-0**	**MR-1.1**	**MR-1.2**	**MR-2.1**	**MR-2.2**	**MR-3.1**	**MR-3.2**	**MR-4.1**	**MR-4.2**	**MR-5.1**	**MR-5.2**
1	+= to -=	×	×	×	×	×	×	×	✓	✓	×	×
2	!= to ==	×	×	×	×	×	×	×	×	×	×	×
3	[i+1] to [i-1]	×	✓	✓	✓	×	✓	×	×	×	×	×
4	+= to -=	✓	×	×	×	×	×	×	×	×	×	×
5	!= to ==	×	×	×	×	×	×	×	×	×	×	×
6	/= to *=	✓	×	×	×	×	×	×	×	×	×	×
7	== to !=	×	×	×	×	✓	×	×	×	✓	×	×
8	-1 to +1	×	×	×	×	×	✓	×	×	×	×	×
9	305: += to -=	×	×	×	×	×	×	×	×	×	×	×
10	307:* to /	×	×	×	×	×	×	×	×	×	×	×
11	317:- to +	×	×	×	×	×	×	×	×	×	×	×
12	348: Delete not	×	×	×	×	×	✓	×	×	×	×	✓
13	353:== to !=	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	✓
14	362: -1 to +1	×	×	×	×	×	×	×	×	×	×	×
15	384: 2 to 3	×	×	×	×	×	×	×	×	×	×	×
16	387:/ to *	×	×	×	×	×	✓	×	×	×	×	×
17	414: removed not	×	×	×	×	×	×	×	×	×	×	×
18	416: REMOVED	×	×	×	×	✓	×	×	×	×	×	×

	NOT											
19	424: == to !=	×	×	×	×	×	×	×	×	×	×	×
20	427: -1 TO 1	×	×	×	×	×	×	×	×	×	×	×
21	445: -to +	×	×	×	×	×	×	×	×	×	×	×
22	450: layer_units[1:] to layer_units[1:10]	×	×	×	✓	×	×	×	×	×	×	×
23	455: [1:] to[1:10]	×	×	×	×	✓	×	×	×	×	×	×
24	566: True to False	×	×	×	×	×	×	×	×	×	×	×
25	566: >= to <=	×	×	×	×	×	×	×	×	×	×	×
26	570: -1 to +1	×	×	×	×	×	×	×	×	×	×	×
27	590: + to -	×	×	×	×	×	×	×	×	×	×	×
28	597: == to !=	×	×	×	×	×	×	×	×	×	×	×
29	607:and to or	×	×	×	×	×	×	×	×	×	×	×
30	619:0 to 1	×	×	×	×	×	×	×	×	×	×	×
31	624: min to max	×	×	×	×	×	×	×	×	×	×	×
32	628: <1 TO >1	×	×	×	×	×	×	×	×	×	×	×
33	628: or to and	×	×	×	×	×	×	×	×	×	×	×
34	636: not added	×	×	×	×	×	×	×	×	×	×	×
35	644: 0.0 TO 1.0	×	×	×	×	×	×	×	×	×	×	×
36	647: not added	×	×	×	×	×	×	×	×	×	×	×
37	655: 0 to 1	×	×	×	×	×	×	×	×	×	×	×
38	662: += TO -=	×	×	×	×	×	×	×	×	×	×	×
39	662: - TO +	×	×	×	×	×	×	×	×	×	×	×
40	662: * to /	×	×	×	×	×	×	×	×	×	×	×
41	667: + to -	×	×	×	×	×	×	×	×	×	×	×
42	671: += TO -=	×	×	×	×	×	×	×	×	×	×	×
43	673 * TO /	×	×	×	×	×	×	×	×	×	×	×
44	675 += TO -=	×	×	×	×	×	×	×	×	×	×	×
45	691: not added	×	×	×	×	×	×	×	×	×	×	×
46	704: not added	×	×	×	×	×	×	×	×	×	×	×
47	706 break to continue	×	×	×	×	×	×	×	×	×	×	×
48	713: == TO !=	×	×	×	×	×	×	×	×	×	×	×
49	739: > TO <	×	×	×	×	×	×	×	×	×	×	×
50	741: += TO -=	×	×	×	×	×	×	×	×	×	×	×
51	742: += to -+	×	×	×	×	×	×	×	×	×	×	×
52	752: [-1] to [1]	×	×	×	×	×	×	×	×	×	×	×
53	755: += to -=	×	×	×	×	×	×	×	×	×	×	×
54	760: [-1] to [1]	×	×	×	×	×	×	×	×	×	×	×
55	1056: max_iter = 200 to max_iter= -200	×	×	×	×	×	×	×	×	×	×	×
56	1082:== TO !=	×	×	×	×	✓	×	×	×	×	×	×
57	1107: removed not	×	×	×	×	✓	×	×	×	×	×	×
58	1107: or to and	×	×	×	×	×	×	×	×	×	×	×
59	1109 removed not	×	×	×	×	×	×	×	×	×	×	×
60	1109 removed second not	×	×	×	×	×	×	×	×	×	×	×
61	1122 != to ==	×	×	×	×	×	×	×	×	×	×	×
62	1158 == TO !=	×	×	×	×	×	×	×	×	×	×	×
63	1189: Removed	×	×	×	×	×	×	×	×	×	×	×

	not											
64	1243: == TO !=	×	×	×	×	×	×	×	×	×	×	×
65	1248 – TO +	×	×	×	×	×	×	×	×	×	×	×
66	1533: == TO !=	×	×	×	×	×	×	×	×	×	×	×
67	1544: == TO !=	×	×	×	×	×	×	×	×	×	×	×
		4%	3%	3%	6%	25%	7%	1%	6%	6%	0%	3%
14/67 = 20.8%												

Table 7: *Mutation Coverage for ANN with Titanic*

As per the detailed coverage report for ANN when it is trained with titanic and then tested with the metamorphic relations MR-0 to MR5.2, 43% of the statements are covered. The kill rate for the metamorphic relations with 67 mutants is 6% where the highest kill rate is 25% by MR-2.2 and the lowest is 0% by MR-5.1. The mutants are generated in such a way that they are evenly distributed throughout the algorithms and every mutant is individually tested multiple times to ensure that the kill is due to the MRs and not due to the misclassification by the model. 38 out of 67 mutants were in the part of the code that is never covered by either statement coverage or branch coverage. Therefore the mutants that are executed are 29 and out of these 29 mutants the mutants that are killed is equal to 14. That brings the kill rate to 48% which is significantly higher that the initial 6%.

Kill rate for the metamorphic relations for ANN with the Iris Dataset												
Sr no	**Mutant**	**MR-0**	**MR-1.1**	**MR-1.2**	**MR-2.1**	**MR-2.2**	**MR-3.1**	**MR-3.2**	**MR-4.1**	**MR-4.2**	**MR-5.1**	**MR-5.2**
1	+= to -=	✓	✓	×	×	×	×	✓	×	✓	×	✓
2	!= to ==	×	×	×	✓	✓	✓	✓	✓	✓	×	×
3	[i+1] to [i-1]	✓	×	×	×	×	×	×	✓	✓	×	×
4	+= to -=	✓	✓	×	✓	×	×	✓	✓	✓	×	✓
5	!= to ==	×	×	✓	×	×	×	×	×	×	×	×
6	/= to *=	✓	×	×	✓	✓	✓	✓	✓	✓	×	✓
7	== to !=	×	×	×	×	×	×	×	×	×	×	×
8	-1 to +1	-	-	-	-	-	-	-	-	-	-	-
9	305: += to -=	×	×	×	×	×	×	×	×	×	×	×
10	307:* to /	✓	✓	×	✓	×	×	×	×	✓	✓	✓
11	317:- to +	✓	✓	✓	×	×	✓	✓	✓	✓	×	✓
12	348: Delete not	×	✓	✓	✓	✓	✓	×	×	✓	✓	✓
13	353:== to !=	✓	✓	✓	✓	✓	×	×	✓	✓	✓	×
14	362: -1 to +1	-	-	-	-	-	-	-	-	-	-	-
15	384: 2 to 3	×	×	×	×	×	×	×	×	×	×	×
16	387:/ to *	✓	×	×	✓	✓	✓	✓	✓	✓	×	✓
17	413: removed not	×	×	×	×	×	×	×	×	×	×	×
18	416: removed not	×	×	×	×	×	×	×	×	×	×	×
19	424: == to !=	×	×	×	×	×	×	×	×	×	×	×
20	427: -1 TO 1	×	×	×	×	×	×	×	×	×	×	×
21	445: -to +	×	×	×	×	×	×	×	×	×	×	×
22	450: layer_units[1:] to layer_units[1:10]	×	×	×	×	×	×	×	×	×	×	×
23	455: [1:] to[1:10]	×	×	×	×	×	×	×	×	×	×	×
24	566: True to False	×	×	×	×	×	×	×	×	×	×	×
25	566: >= to <=	×	×	×	×	×	×	×	×	×	×	×
26	570: -1 to +1	×	×	×	×	×	×	×	×	×	×	×

27	590: + to -	×	×	×	×	×	×	×	×	×	×	×
28	597: == to !=	×	×	×	×	×	×	×	×	×	×	×
29	607:and to or	×	×	×	×	×	×	×	×	×	×	×
30	619:0 to 1	×	×	×	×	×	×	×	×	×	×	×
31	624: min to max	×	×	×	×	×	×	×	×	×	×	×
32	628: <1 TO >1	×	×	×	×	×	×	×	×	×	×	×
33	628: or to and	×	×	×	×	×	×	×	×	×	×	×
34	636: not added	×	×	×	×	×	×	×	×	×	×	×
35	644: 0.0 TO 1.0	×	×	×	×	×	×	×	×	×	×	×
36	647: not added	×	×	×	×	×	×	×	×	×	×	×
37	655: 0 to 1	×	×	×	×	×	×	×	×	×	×	×
38	662: += TO -=	×	×	×	×	×	×	×	×	×	×	×
39	662: - TO +	×	×	×	×	×	×	×	×	×	×	×
40	662: * to /	×	×	×	×	×	×	×	×	×	×	×
41	667: + to -	×	×	×	×	×	×	×	×	×	×	×
42	671: += TO -=	×	×	×	×	×	×	×	×	×	×	×
43	673 * TO /	×	×	×	×	×	×	×	×	×	×	×
44	675 += TO -=	×	×	×	×	×	×	×	×	×	×	×
45	692: not added	×	×	×	×	×	×	×	×	×	×	×
46	704: not added	×	×	×	×	×	×	×	×	×	×	×
47	706 break to continue	×	×	×	×	×	×	×	×	×	×	×
48	713: == TO !=	×	×	×	×	×	×	×	×	×	×	×
49	739: > TO <	×	×	×	×	×	×	×	×	×	×	×
50	741: += TO -=	×	×	×	×	×	×	×	×	×	×	×
51	742: += to -=	×	×	×	×	×	×	×	×	×	×	×
52	752: [-1] to [1]	×	×	×	×	×	×	×	×	×	×	×
53	755: += to -=	×	×	×	×	×	×	×	×	×	×	×
54	760: [-1] to [1]	×	×	×	×	×	×	×	×	×	×	×
55	1056: max_iter = 200 to max_iter= -200	×	×	×	×	×	×	×	×	×	×	×
56	1082: == TO!=	×	×	×	×	×	×	×	×	×	×	×
57	1107: removed not	×	×	×	×	×	×	×	×	×	×	×
58	1107: or to and	×	×	×	×	×	×	×	×	×	×	×
59	1109 removed not	×	×	×	×	×	×	×	×	×	×	×
60	1109 removed second not	×	×	×	×	×	×	×	×	×	×	×
61	1122 != to ==	×	×	×	×	×	×	×	×	×	×	×
62	1158 == TO !=	×	×	×	×	×	×	×	×	×	×	×
63	1189: Removed not	×	×	×	×	×	×	×	×	×	×	×
64	1243: == TO !=	×	×	×	×	×	×	×	×	×	×	×
65	1248 – TO +	×	×	×	×	×	×	×	×	×	×	×
66	1533: == TO !=	×	×	×	×	×	×	×	×	×	×	×
67	1544: == TO !=	×	×	×	×	×	×	×	×	×	×	×
		18%	15%	12%	16%	13%	13%	15%	16%	21%	12%	16%
13/67 = 19.4%												

Table 8: *Mutation Coverage for ANN with Iris*

ANN was trained with titanic to see if the results are any different. The same mutants were added at the same points again to see if the results are same with a different dataset. 36 mutants in this case were in the uncovered part of the algorithm, which makes the mutants that were covered and killed 41.9%. its

interesting to not that some of the mutants that were in the uncovered part with iris were in the covered part with titanic.

Kill rate for the metamorphic relations for ID3 with the Iris Dataset No of lines of Code: 1833												
Sr no	**Mutant**	**MR-0**	**MR-1.1**	**MR-1.2**	**MR-2.1**	**MR-2.2**	**MR-3.1**	**MR-3.2**	**MR-4.1**	**MR-4.2**	**MR-5.1**	**MR-5.2**
1	132: 0.0 to 1.0	×	×	×	×	×	×	×	×	×	×	×
2	161: added not	×	×	×	×	×	×	×	×	×	×	×
3	172: added not	×	×	×	×	×	×	×	×	×	×	×
4	180 < to >	×	×	×	×	×	×	×	×	×	×	×
5	188: += to -+	×	×	×	×	×	×	×	×	×	×	×
6	203: or to and	×	×	×	×	×	×	×	×	×	×	×
7	203: != to ==	×	×	×	×	×	×	×	×	×	×	×
8	208: < to >	×	×	×	×	×	×	×	×	×	×	×
9	204: <= to >=	×	×	×	×	×	×	×	×	×	×	×
10	237: y to y+1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
11	242: removed not	×	×	×	×	×	×	×	×	×	×	×
12	259: != to ==	×	×	×	×	×	×	×	×	×	×	×
13	277: > to <	×	×	×	×	×	×	×	×	×	×	×
14	277: <= to >=	×	×	×	×	×	×	×	×	×	×	×
15	283: * to /	×	×	×	×	×	×	×	×	×	×	×
16	287: >= to <=	×	×	×	×	×	×	×	×	×	×	×
17	294: < to >	×	×	×	×	×	×	×	×	×	×	×
18	301:max to min	×	×	×	×	×	×	×	×	×	×	×
19	304: max to min	×	×	×	×	×	×	×	×	×	×	×
20	206: * to /	×	×	×	×	×	×	×	×	×	×	×
21	312: max to min	×	×	×	×	×	×	×	×	×	×	×
22	316: == to !=	×	×	×	×	×	×	×	×	×	×	×
23	318: max to min	×	×	×	×	×	×	×	×	×	×	×
24	321: max to min	×	×	×	×	×	×	×	×	×	×	×
25	332:> to <	×	×	×	×	×	×	×	×	×	×	×
26	345: <=to >=	×	×	×	×	×	×	×	×	✓	×	×
27	360: <2 to >2	×	×	×	×	×	×	×	×	×	×	×
28	365: removed not	×	×	×	×	×	×	×	×	×	×	×
29	369: removed not	×	×	×	×	×	×	×	×	✓	×	×
30	372: removed not	×	×	×	×	×	×	×	×	×	×	×
31	382: * to /	×	×	×	×	×	×	×	×	×	×	×
32	386:* to /	×	×	×	×	×	×	×	×	×	×	×
33	400: < to >	×	×	×	×	×	×	×	×	×	×	×
34	408: < to >	×	×	×	×	×	×	×	×	×	×	×
35	415: != to ==	✓	×	×	×	×	×	×	×	×	×	×
36	427: removed not	×	×	×	×	×	×	×	×	×	×	×
37	461: 1 to 0	×	×	×	×	×	×	×	×	×	×	×
38	462: * to /	×	×	×	×	×	×	×	×	×	×	×

39	487: and to or	×	×	×	×	×	×	×	×	×	×	×
40	502: False to True	×	×	×	×	×	×	×	×	×	×	×
41	507: or to and	×	×	×	×	×	×	×	×	×	×	×
42	508: != to ==	✓	×	×	×	×	×	×	×	×	×	×
43	515: False to True	✓	×	×	×	×	×	×	×	×	×	×
44	545 x to X+1	×	×	×	×	×	×	×	×	×	×	×
45	550: added not	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
46	571: == to !=	×	×	×	×	×	×	×	×	×	×	×
47	636:> to <	✓	×	×	×	×	×	×	×	×	×	×
48	641 == to !=	✓	×	×	×	×	×	×	×	×	×	×
49	654 * to /	×	×	×	×	×	×	×	×	×	×	×
50	980: True to False	×	×	×	×	×	×	×	×	×	×	×
51	1055: == to !=	×	×	×	×	×	×	×	×	×	×	×
52	1057: /= to *=	×	×	×	×	×	×	×	×	×	×	×
53	1069: == to !=	×	×	×	×	×	×	×	×	×	×	×
54	1071: /= to *=	×	×	×	×	×	×	×	×	×	×	×
55	1096: == to !=	×	×	×	×	×	×	×	×	×	×	×
56	1849: 2 to 3	×	×	×	×	×	×	×	×	×	×	×
		20%	4%	4%	4%	4%	4%	4%	4%	13 %	4%	4%
9/56 = 16.07%												

Table 9: *Mutation Coverage for ID3 with Iris*

Iris is one of the most popular datasets for classification, because of its high accuracy and the quality of being balanced. Very little pre- processing is needed to work with Iris. ID3 was trained with Iris and overall kill rate for ID3 with Iris was 14%, since 8 out of 56 mutants were killed by test suite on the whole. The statement and branch coverage with this dataset and ID3 are 46.7% and 5%. The code for ID3 is different from the others in the fact that it has a large number of nested conditional statements. The covered and uncovered parts are highly segregated because of the multiple conditions and the parts within them that remain uncovered.

Kill rate for the metamorphic relations for ID3 with the Titanic Dataset No of lines of Code: 1833												
Sr no	**Mutant**	**MR-0**	**MR-1.1**	**MR-1.2**	**MR-2.1**	**MR-2.2**	**MR-3.1**	**MR-3.2**	**MR-4.1**	**MR-4.2**	**MR-5.1**	**MR-5.2**
1	132: 0.0 to 1.0	×	×	×	×	×	×	×	×	×	×	×
2	161: added not	×	×	×	×	×	×	×	×	×	×	×
3	172: added not	×	×	×	×	×	×	×	×	×	×	×
4	180 < to >	×	×	×	×	×	×	×	×	×	×	×
5	188: += to -=	×	×	×	×	×	×	×	×	×	×	×
6	203: or to and	×	×	×	×	×	×	×	×	×	×	×
7	203: != to ==	×	×	×	×	×	×	×	×	×	×	×
8	208: < to >	×	×	×	×	×	×	×	×	×	×	×
9	214: <= to >=	×	×	×	×	×	×	×	×	×	×	×
10	237: y to y+1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
11	242: removed not	✓	×	×	×	×	×	×	×	×	×	×
12	259: != to ==	×	×	×	×	×	×	×	×	×	×	×
13	277: > to <	×	×	×	×	×	×	×	×	×	×	×
14	277: <= to >=	×	×	×	×	×	×	×	×	×	×	×
15	283: * to /	×	×	×	×	×	×	×	×	×	×	×

16	287: >= to <=	×	×	×	×	×	×	×	×	×	×	×
17	294: < to >	×	×	×	×	×	×	×	×	×	×	×
18	301:max to min	×	×	×	×	×	×	×	×	×	×	×
19	304: max to min	×	×	×	×	×	×	×	×	×	×	×
20	206: * to /	×	×	×	×	×	×	×	×	×	×	×
21	312: max to min	×	×	×	×	×	×	×	×	×	×	×
22	316: == to !=	×	×	×	×	×	×	×	×	×	×	×
23	318: max to min	×	×	×	×	×	×	×	×	×	×	×
24	321: max to min	×	×	×	×	×	×	×	×	×	×	×
25	332:> to <	×	×	×	×	×	×	×	×	×	×	×
26	345: <=to >=	×	×	×	×	×	×	×	×	×	×	×
27	360: <2 to >2	×	×	×	×	×	×	×	×	×	×	×
28	365: removed not	✓	×	×	×	×	×	×	×	×	×	×
29	369: removed not	×	×	×	×	×	×	×	×	×	×	×
30	372: removed not	×	×	×	×	×	×	×	×	×	×	×
31	382: * to /	×	×	×	×	×	×	×	×	×	×	×
32	386:* to /	×	×	×	×	×	×	×	×	×	×	×
33	400: < to >	×	×	×	×	×	×	×	×	×	×	×
34	408: < to >	×	×	×	×	×	×	×	×	×	×	×
35	415: != to ==	×	×	×	×	×	×	×	×	×	×	×
36	427: removed not	✓	×	×	×	×	×	×	×	×	×	×
37	461: 1 to 0	×	×	×	×	×	×	×	×	×	×	×
38	462: * to /	×	×	×	×	×	×	×	×	×	×	×
39	487: and to or	×	×	×	×	×	×	×	×	×	×	×
40	502: False to True	×	×	×	×	×	×	×	×	×	×	×
41	507: or to and	×	×	×	×	×	×	×	×	×	×	×
42	508: != to ==	×	×	×	×	×	×	×	×	×	×	×
43	515: False to True	×	×	×	×	×	×	×	×	×	×	×
44	545 x to X+1	×	×	×	×	×	×	×	×	×	×	×
45	550: added not	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
46	571: == to !=	×	×	×	×	×	×	×	×	×	×	×
47	636:> to <	×	×	×	×	×	×	×	×	×	×	×
48	641 == to !=	✓	×	×	×	×	×	×	×	×	×	×
49	654 * to /	×	×	×	×	×	×	×	×	×	×	×
50	980: True to False	×	×	×	×	×	×	×	×	×	×	×
51	1055: == to !=	×	×	×	×	×	×	×	×	×	×	×
52	1057: /= to *=	×	×	×	×	×	×	×	×	×	×	×
53	1069: == to !=	×	×	×	×	×	×	×	×	×	×	×
54	1071: /= to *=	×	×	×	×	×	×	×	×	×	×	×
55	1096: == to !=	×	×	×	×	×	×	×	×	×	×	×
56	1849: 2 to 3	×	×	×	×	×	×	×	×	×	×	×
		14%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%
6/56 = 10.71%												

Table 10: *Mutation Coverage for ID3 with Titanic*

To verify that the kill rate and coverage value are due to the MRs and are not affected by the choice of the algorithm or the dataset being used, ID3 was trained with Titanic and 5 out of 56 mutants were killed. 23 were not covered and 28 survived despites being covered. This makes the kill rate for the mutants that were in the covered part 16%. As it is apparent from the above table the covered parts here as well are segregated.

Kill rate for the metamorphic relations for SVM with the Titanic Dataset No of lines of Code: 1496												
Sr	**Mutant**	**MR-**	**MR-**	**MR-**	**MR-**	**MR-**	**MR-**	**MR-**	**MR-**	**MR-**	**MR-**	**MR-**

no		0	1.1	1.2	2.1	2.2	3.1	3.2	4.1	4.2	5.1	5.2
1	211: Verbose = 0 to verbose=1	×	×	×	×	×	×	×	×	×	×	×
2	250: < to >	×	×	×	×	×	×	×	×	×	×	×
3	158: False to True	×	×	×	×	×	×	×	×	×	×	×
4	269: and to or	×	×	×	×	×	×	×	×	×	×	×
5	270: == to !=	×	×	×	×	×	×	×	×	×	×	×
6	217: -1 to 1	×	×	×	×	×	×	×	×	×	×	×
7	275: 0 to 1	×	×	×	×	×	×	×	×	×	×	×
8	450: < to >	×	×	×	×	×	×	×	×	×	×	×
9	457: False to True	×	×	×	×	×	×	×	×	×	×	×
10	687: 1e -3 to 1e+3	×	×	×	×	×	×	×	×	×	×	×
11	909: -1 to 1	×	×	×	×	×	×	×	×	×	×	×
12	1077: -1 to 1	×	×	×	×	×	×	×	×	×	×	×
13	1247: -1 to 1	×	×	×	×	×	×	×	×	×	×	✓
14	1386: False to True	×	×	×	×	×	×	×	×	×	×	×
15	1451: + to -	×	×	×	×	×	×	×	×	×	×	×
		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	6%
1/15= 6.66%												

Table 11: *Mutation Coverage for SVM with Titanic*

The number of mutants in SVM are a lot less than the other algorithms, although the lines of code is big. This reason for this is that the code contained very few mutant operators that could be modified, which limits the number of mutants that can be added to the code. Another reason is that the code was largely composed of function calls and other language specific constraints which cannot be easily mutated. 10 out of 15 mutants are in the area that was not covered, resulting in 1 mutant being killed and 4 mutants were covered and they survived. The kill rate for the covered mutants is 20%.

Kill rate for the metamorphic relations for SVM with the Iris Dataset No of lines of Code: 1496												
Sr no	**Mutant**	**MR-0**	**MR-1.1**	**MR-1.2**	**MR-2.1**	**MR-2.2**	**MR-3.1**	**MR-3.2**	**MR-4.1**	**MR-4.2**	**MR-5.1**	**MR-5.2**
1	211: Verbose = 0 to verbose=1	×	×	×	×	×	×	×	×	×	×	×
2	250: < to >	×	×	×	×	×	×	×	×	×	×	×
3	158: False to True	×	×	×	×	×	×	×	×	×	×	×
4	269: and to or	×	×	×	×	×	×	×	×	×	×	×
5	270: == to !=	×	×	×	×	×	×	×	×	×	×	×
6	217: -1 to 1	×	×	×	×	×	×	×	×	×	×	×
7	275: 0 to 1	×	×	×	×	×	×	×	×	×	×	×
8	450: < to >	×	×	×	×	×	×	×	×	×	×	×
9	457: False to True	×	×	×	×	×	×	×	×	×	×	×
10	687: 1e -3 to 1e+3	×	×	×	×	×	×	×	×	×	×	×
11	909: -1 to 1	×	×	×	×	×	×	×	×	×	×	×
12	1077: -1 to 1	×	×	×	×	×	×	×	×	×	×	×
13	1247: -1 to 1	×	×	×	×	×	×	×	×	×	×	✓
14	1386: False to True	×	×	×	×	×	×	×	×	×	×	×
15	1451: + to -	×	×	×	×	×	×	×	×	×	×	×
		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	6%
1/15 = 6.66%												

Table 12: *Mutation Coverage for SVM with Iris*

The code for KNN has 382 lines which is relatively less than the other algorithms and there is a lack of mutants in the code as well so there are only 11 mutants for it. Only three mutants out of these 11 were covered and no mutants were killed for either Iris or Titanic.

Kill rate for the metamorphic relations for KNN with the Iris Dataset No of lines of Code: 382												
Sr no	Mutant	MR-0	MR-1.1	MR-1.2	MR-2.1	MR-2.2	MR-3.1	MR-3.2	MR-4.1	MR-4.2	MR-5.1	MR-5.2
1	185: 5 to -5	×	×	×	×	×	×	×	×	×	×	×
2	239: X to X+1	×	×	×	×	×	×	×	×	×	×	×
3	258: X to X+1	×	×	×	×	×	×	×	×	×	×	×
4	284: removed not	×	×	×	×	×	×	×	×	×	×	×
5	318: Removed not	×	×	×	×	×	×	×	×	×	×	×
6	322: -1 to 1	×	×	×	×	×	×	×	×	×	×	×
7	neigh_ind to neigh_ind +1	×	×	×	×	×	×	×	×	×	×	×
8	358: : to :10	×	×	×	×	×	×	×	×	×	×	×
9	366 == to !=	×	×	×	×	×	×	×	×	×	×	×
10	370: /= to *=	×	×	×	×	×	×	×	×	×	×	×
11	357: 0 to 1	×	×	×	×	×	×	×	×	×	×	×
		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
0/11 = 0%												

Table 13: *Mutation Coverage for KNN with Iris*

Kill rate for the metamorphic relations for KNN with the Titanic Dataset No of lines of Code: 382												
Sr no	**Mutant**	**MR-0**	**MR-1.1**	**MR-1.2**	**MR-2.1**	**MR-2.2**	**MR-3.1**	**MR-3.2**	**MR-4.1**	**MR-4.2**	**MR-5.1**	**MR-5.2**
1	185: 5 to -5	×	×	×	×	×	×	×	×	×	×	×
2	239: X t0 X+1	×	×	×	×	×	×	×	×	×	×	×
3	258: X to X+1	×	×	×	×	×	×	×	×	×	×	×
4	284: removed not	×	×	×	×	×	×	×	×	×	×	×
5	318: Removed not	×	×	×	×	×	×	×	×	×	×	×
6	322: -1 to 1	×	×	×	×	×	×	×	×	×	×	×
7	neigh_ind to neigh_ind +1	×	×	×	×	×	×	×	×	×	×	×
8	358: : to :10	×	×	×	×	×	×	×	×	×	×	×
9	366 == to !=	×	×	×	×	×	×	×	×	×	×	×
10	370: /= to *=	×	×	×	×	×	×	×	×	×	×	×
11	357: 0 to 1	×	×	×	×	×	×	×	×	×	×	×
		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
0/11 = 0%												

Table 14: *Mutation Coverage for KNN with Titanic*

Kill rate for the metamorphic relations for Naïve Bayes with the

Iris Dataset No of lines of Code: 522												
Sr no	**Mutant**	**MR-0**	**MR-1.1**	**MR-1.2**	**MR-2.1**	**MR-2.2**	**MR-3.1**	**MR-3.2**	**MR-4.1**	**MR-4.2**	**MR-5.1**	**MR-5.2**
1	147: X to X+1	×	×	×	×	×	×	×	×	×	×	×
2	152: - to +	×	×	×	×	×	×	×	×	×	×	×
3	235: 1e-9 to 1e+9	✓	✓	×	✓	✓	✓	×	✓	✓	✓	×
4	264: True to False	✓	✓	×	✓	✓	✓	×	✓	✓	✓	×
5	266: True to False	✓	✓	×	✓	✓	✓	×	✓	✓	✓	×
6	313: == to !=	✓	✓	×	✓	✓	✓	×	✓	✓	✓	×
7	323: 2 to 3	×	×	×	×	×	×	×	×	×	×	×
8	328: 0 to 1	×	×	×	×	×	×	×	×	×	×	×
9	333: == to !=	×	×	×	×	×	×	×	×	×	×	×
10	338: + to -	×	×	×	×	×	×	×	×	×	×	×
11	345: n_new * to n_new /	×	×	×	×	×	×	×	×	×	×	×
12	345: n_new + to -	×	×	×	×	×	×	×	×	×	×	×
13	346: / to *	×	×	×	×	×	×	×	×	×	×	×
14	353: * to /	×	×	×	×	×	×	×	×	×	×	×
15	355: * to /	×	×	×	×	×	×	×	×	×	×	×
16	357: + to -	×	×	×	×	×	×	×	×	×	×	×
17	359:* to /	×	×	×	×	×	×	×	×	×	×	×
18	406: False to True	×	×	×	×	×	×	×	×	×	×	×
19	438: Removed not	×	×	×	×	×	×	×	×	×	×	×
20	449: * to /	×	×	×	×	×	×	×	×	×	×	×
21	461: 0 to 1	×	×	×	×	×	×	×	×	×	×	×
22	475: != to ==	×	×	×	×	×	×	×	×	×	×	×
23	481: Removed not	×	×	×	×	×	×	×	×	×	×	×
24	486: < to >	×	×	×	×	×	×	×	×	×	×	×
25	496 != to ==	×	×	×	×	×	×	×	×	×	×	×
26	502 : to :10	×	×	×	×	×	×	×	×	×	×	×
27	518: == to !=	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
28	522: == to !=	×	×	×	×	×	×	×	×	×	×	×
29	527: 0 to 1	×	×	×	×	×	×	×	×	×	×	×
30	531: : to :10	×	×	×	×	×	×	×	×	×	×	×
31	534: : to :10	×	×	×	×	×	×	×	×	×	×	×
32	537: += to -=	×	×	×	×	×	×	×	×	×	×	×
33	540: += to -=	×	×	×	×	×	×	×	×	✓	×	×
34	547: / to *	×	×	×	×	×	×	×	×	×	×	×
35	557: - to +	×	×	×	×	×	×	×	×	×	×	×
36	560:* to /	×	×	×	×	×	×	×	×	×	×	×
37	560: - to +	×	✓	×	✓	✓	✓	✓	✓	✓	✓	✓
38	563: + to -	✓	×	×	×	×	×	×	×	✓	×	×
		16%	16%	3%	16%	16%	16%	5%	16%	21%	16%	5%
8/38 = 21.05%												

Table 15: *Mutation Coverage for Naïve Bayes with Iris*

8 mutants were killed out of 38 mutants that were added to Naïve Bayes. 16 mutants were in the uncovered parts of the code. Naïve Bayes had a code coverage of 46.4 and 31.5 % for statement and branch coverage and the overall kill rate for the MRs was 21% for Iris and 7.8% for Titanic.

Kill rate for the metamorphic relations for Naïve Bayes with the Titanic Dataset No of lines of Code: 522												
Sr no	**Mutant**	**MR-0**	**MR-1.1**	**MR-1.2**	**MR-2.1**	**MR-2.2**	**MR-3.1**	**MR-3.2**	**MR-4.1**	**MR-4.2**	**MR-5.1**	**MR-5.2**
1	147: X to X+1	×	×	×	×	×	×	×	×	×	×	×
2	152: - to +	×	×	×	×	×	×	×	×	×	×	×
3	235: 1e-9 to 1e+9	×	×	×	×	×	×	×	✓	x	x	x
4	264: True to False	×	×	×	×	×	×	×	×	×	×	×
5	266: True to False	×	×	×	×	×	×	×	×	×	×	×
6	313: == to !=	×	×	×	×	×	×	×	✓	x	x	x
7	323: 2 to 3	×	×	×	×	×	×	×	×	×	×	×
8	328: 0 to 1	×	×	×	×	×	×	×	×	×	×	×
9	340: == to !=	×	×	×	×	×	×	×	×	×	×	×
10	344: + to -	×	×	×	×	×	×	×	×	×	×	×
11	352: n_new * to n_new /	×	×	×	×	×	×	×	×	×	×	×
12	352: n_new + to -	×	×	×	×	×	×	×	×	×	×	×
13	352: / to *	×	×	×	×	×	×	×	×	×	×	×
14	359: * to /	×	×	×	×	×	×	×	×	×	×	×
15	361: * to /	×	×	×	×	×	×	×	×	×	×	×
16	363: + to -	×	×	×	×	×	×	×	×	×	×	×
17	365:* to /	×	×	×	×	×	×	×	×	×	×	×
18	412: False to True	×	×	×	×	×	×	×	×	×	×	×
19	443: Removed not	-	-	-	-	-	-	-	-	-	-	-
20	454: * to /	×	×	×	×	×	×	×	×	×	×	×
21	461: 0 to 1	-	-	-	-	-	-	-	-	-	-	-
22	478: != to ==	×	×	×	×	×	×	×	×	×	×	×
23	485: Removed not	×	×	×	×	×	×	×	×	×	×	×
24	489: < to >	×	×	×	×	×	×	×	×	×	×	×
25	500 != to ==	×	×	×	×	×	×	×	×	×	×	×
26	506 : to :10	×	×	×	×	×	×	×	×	×	×	×
27	521: == to !=	✓	✓	✓	✓	✓	✓	✓	✓	✓	x	✓
28	526: == to !=	×	×	×	×	×	×	×	×	×	×	×
29	531: 0 to 1	×	×	×	×	×	×	×	×	×	×	×
30	535: : to :10	×	×	×	×	×	×	×	×	×	×	×
31	538: : to :10	×	×	×	×	×	×	×	×	×	×	×
32	541: += to -=	×	×	×	×	×	×	×	×	×	×	×
33	544: += to -=	×	×	×	×	×	×	×	×	×	×	×
34	550: / to *	×	×	×	×	×	×	×	×	×	×	×
35	559: - to +	×	×	×	×	×	×	×	×	×	×	×
36	563:* to /	×	×	×	×	×	×	×	×	×	×	×
37	563: - to +	×	×	×	×	×	×	×	×	×	×	×
38	563: + to -	×	×	×	×	×	×	×	×	×	×	×
		8%	8%	8%	8%	8%	8%	8%	13%	8%	5%	8%
3/38 = 7.89%												

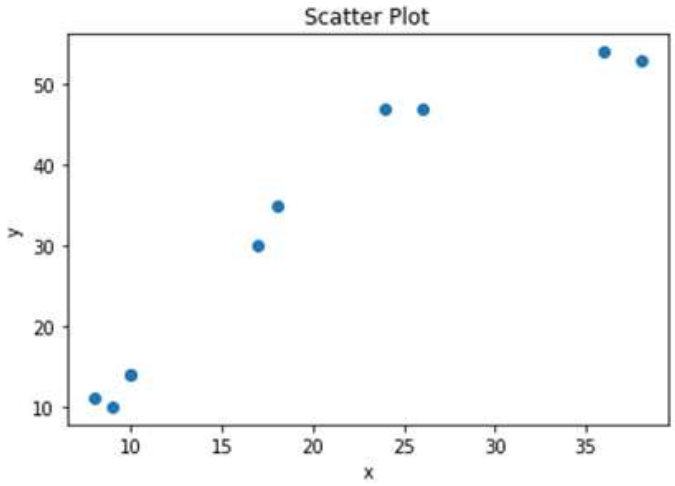
Table 16: *Mutation Coverage for Naïve Bayes with Titanic*

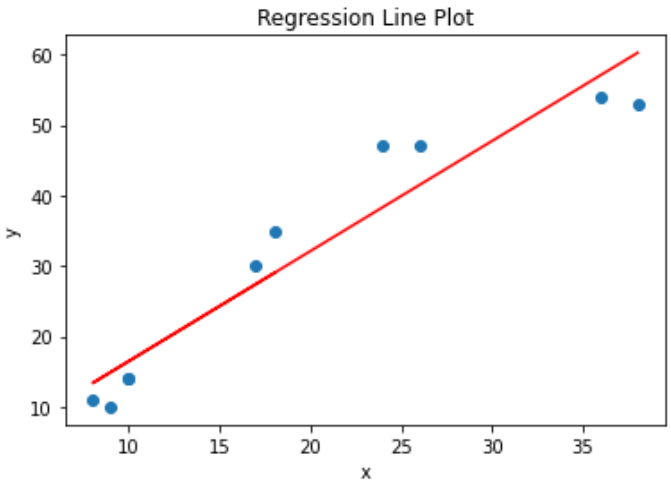
IV. ANALYSIS & FINDINGS

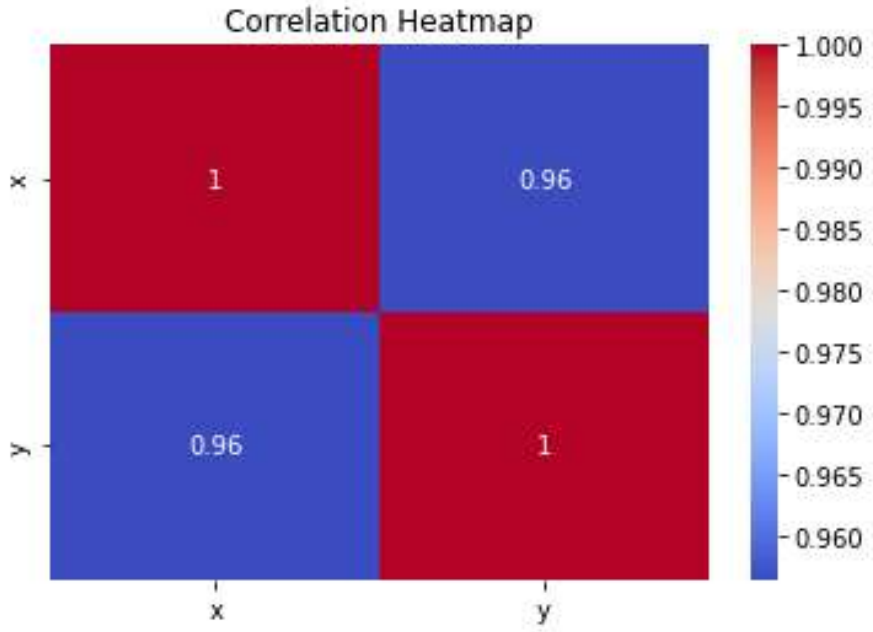
It is clear from the above discussion that more than 50% when testing with Xie's MR's don't get detected at all because they lie in the part of the code that is never exercised either by statement or branch coverage. Although the kill rate of the given MRs is independent the code covered, because the kill rate is affected by a complex set of factors like the quality of the test cases, the type and the diversity of the mutants used and the effectiveness of the

test cases. The fact still holds that if the mutants that are in the uncovered part of the code could get executed, it is possible that some of the mutants might get killed and increase the overall kill rate of the test Suite.

We found a strong correlation between the number of mutants that were not killed and the number of mutants that were not covered. To see if this correlation held statistically, the data form the kill rate and the coverage was used to find the Pearson's Correlation and the P-Value for the number of uncovered mutants and the number of mutants that were not killed. The Pearson's Correlation coefficient came out to: 0.956453459441019 and the P-Value is 1.4925035439266146e-05. This value shows a strong correlation between the two categories and the P-value shows statistically that the likelihood of the correlations to have occurred by chance is very low. The graphs below show the visual representation of the findings







Which means that if the number of uncovered mutants is reduced, it may result in the reduction of the number of mutants that have survived.

V. CONCLUSION

This study quantifies the effectiveness of detecting faults of

Xie's metamorphic relations (MRs) by finding the kill rate for the MRs after applying them to the classifiers from Sci-kit learn library from Python. Along with this, statement and branch coverage were measured before and after applying the MRs. It was observed that there is a negligible impact of MRs on the coverage of the code, which implies that there exists unchecked code when testing with MRs. Another finding is the existence of a robust positive correlation between the uncovered mutants and the ones that survived. This observation suggests that the MRs may exhibit limitations in effectively identifying a broad spectrum of faults. Consequently, future research should focus on incorporating coverage as a metric to enhance the efficacy of testing ML algorithms.

VI. FUTURE WORK

Future work can build upon the insights provided by this work by taking into consideration coverage as one of the metrics for testing Machine Learning classifiers, to address the limitations inherent in the MRs. This makes it necessary to develop innovative techniques and methodologies that can ensure the coverage of the code along with Metamorphic testing. Automated generation of MRs that target uncovered parts of the code can be instrumental in this regard. Further investigation into the impact of coverage-based testing on the overall effectiveness of the MRs is also needed to better understand intricacies of using coverage as a metric for metamorphic testing in different domains and areas. Furthermore ,work to rectify the coverage limitation of MR might result in optimizing the MRs and make them more effective.

REFERENCES

[1] Bai, T., et al. *Metamorphic Testing for Traffic Light Recognition in Autonomous Driving Systems*. in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. 2021. IEEE.

[2] Deng, Y., et al. *BMT: Behavior driven development-based metamorphic testing for autonomous driving models*. in *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)*. 2021. IEEE.

[3] Deng, Y., et al., *A Declarative Metamorphic Testing Framework for Autonomous Driving*. 2022.

[4] Han, J.C. and Z.Q. Zhou. *Metamorphic fuzz testing of autonomous vehicles*. in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 2020.

[5] Pan, Y., H. Ao, and Y. Fan. *Metamorphic Testing for Autonomous Driving Systems in Fog based on Quantitative Measurement*. in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. 2021. IEEE.

[6] Valle, P. *Metamorphic testing of autonomous vehicles: A case study on Simulink*. in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. 2021. IEEE.

[7] Ayerdi, J., et al., *Performance-driven metamorphic testing of cyber-physical systems.* 2022.

[8] Iqbal, M., et al., *Metamorphic testing of Advanced Driver-Assistance System (ADAS) simulation platforms: Lane Keeping Assist System (LKAS) case studies.* 2023. **155**: p. 107104.

[9] Asyrofi, M.H., et al. *Crossasr: Efficient differential testing of automatic speech recognition via text-to-speech*. in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2020. IEEE.

[10] Yoo, S. Metamorphic testing of stochastic optimisation. in 2010 Third International Conference on Software Testing, Verification, and Validation Workshops. 2010. IEEE.

[11] Panichella, A., F.M. Kifetew, and P.J.I.T.o.S.E. Tonella, Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets. 2017. **44**(2): p. 122-158.

[12] Guderlei, R. and J. Mayer. *Statistical metamorphic testing testing programs with random output by means of statistical hypothesis tests and metamorphic testing*. in *Seventh International Conference on Quality Software (QSIC 2007)*. 2007. IEEE.

[13] Xie, X., et al., *Testing and validating machine learning classifiers by metamorphic testing.* 2011. **84**(4): p. 544-558.

[14] Canizares, P.C., A. Núñez, and J.J.E.S.w.A. de Lara, *An expert system for checking the correctness of memory systems using simulation and metamorphic testing.* 2019. **132**: p. 44-62.

[15] Khokhar, M.N., et al., *Metamorphic Testing of AI-based Applications: A Critical Review.* 2020. **11**(4).

[16] Asyrofi, M.H., et al., *Biasfinder: Metamorphic test generation to uncover bias for sentiment analysis systems.* 2021. **48**(12): p. 5087-5101.

[17] Jin, L., Z. Ding, and H.J.M. Zhou, Evaluation of Chinese Natural Language Processing System Based on Metamorphic Testing. 2022. **10**(8): p. 1276.

[18] Jiang, M., et al., *On the effectiveness of testing sentiment analysis systems with metamorphic testing.* 2022. **150**: p. 106966.

[19] Manino, E., et al., *Systematicity, Compositionality and Transitivity of Deep NLP Models: a Metamorphic Testing Perspective.* 2022.

[20] Fan, M., et al. *One step further: evaluating interpreters using metamorphic testing*. in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2022.

[21] Tu, K., M. Jiang, and Z.J.M. Ding, *A metamorphic testing approach for assessing question answering systems.* 2021. **9**(7): p. 726.

[22] Sun, Z., et al. Automatic testing and improvement of machine translation. in Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. 2020.

[23] He, P., C. Meister, and Z. Su. Testing machine translation via referential transparency. in 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). 2021. IEEE.

[24] Liu, Z., Y. Feng, and Z. Chen. DialTest: Automated testing for recurrent-neural-network-driven dialogue systems. in Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2021.

[25] Cao, J., et al., SemMT: a semantic-based testing approach for machine translation systems. 2022. **31**(2): p. 1-36.

[26] Hilmi Asyrofi, M., et al., *BiasFinder: Metamorphic Test Generation to Uncover Bias for Sentiment Analysis Systems.* 2021: p. arXiv: 2102.01859.

[27] Ma, P., S. Wang, and J. Liu. *Metamorphic Testing and Certified Mitigation of Fairness Violations in NLP Models*. in *IJCAI*. 2020.

[28] Khoo, L.S., et al. *Exploring and repairing gender fairness violations in word embedding-based sentiment analysis model through adversarial patches*. in *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 2023. IEEE.

[29] Yuan, Y., et al. *Perception matters: Detecting perception failures of vqa models using metamorphic testing*. in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.

[30] Yang, Z., M.H. Asyrofi, and D. Lo. BiasRV: Uncovering biased sentiment predictions at runtime. in Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2021.

[31] Yang, Z., et al. BiasHeal: On-the-fly black-box healing of bias in sentiment analysis systems. in 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME). 2021. IEEE.

[32] Zhang, L., Y. Zhang, and M. Zhang. Efficient white-box fairness testing through gradient search. in Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2021.

[33] Perera, A., et al., *Search-based fairness testing for regression-based machine learning systems.* 2022. **27**(3): p. 79.

[34] Chen, Z., et al. *MAAT: a novel ensemble approach to addressing fairness and performance bugs for machine learning software*. in *Proceedings of the*

30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2022.

[35] Carvalho, D.S., et al., Montague semantics and modifier consistency measurement in neural language models. 2022.

[36] Murphy, C., G.E. Kaiser, and L. Hu, *Properties of machine learning applications for use in metamorphic testing.* 2008.

[37] Chekam, T.T., et al. *An empirical study on mutation, statement and branch coverage fault revelation that avoids the unreliable clean program assumption*. in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. 2017. IEEE.

[38] Luu, Q.-H., et al., *Testing multiple linear regression systems with metamorphic testing.* 2021. **182**: p. 111062.

[39] Peng, Z., U. Kanewala, and N. Niu. *Contextual understanding and improvement of metamorphic testing in scientific software development*. in *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 2021.

[40] Baxter, I.J.I.A.T.I.S.h.t.c.s.d.f.a.X.f.p., Branch Coverage For Arbitrary Languages Made Easy: Transformation Systems to the Rescue. 2001.

[41] Santelices, R., et al. *Lightweight fault-localization using multiple coverage types*. in *2009 IEEE 31st International Conference on Software Engineering*. 2009. IEEE.

[42] Abraham, A., et al., *Machine learning for neuroimaging with scikit-learn.* 2014: p. 14.

[43] Hackeling, G., *Mastering Machine Learning with scikit-learn*. 2017: Packt Publishing Ltd.

[44] Hao, J., T.K.J.J.o.E. Ho, and B. Statistics, Machine learning made easy: a review of scikit-learn package in python programming language. 2019. **44**(3): p. 348-361.

[45] Kramer, O. and O.J.M.l.f.e.s. Kramer, *Scikit-learn.* 2016: p. 45-53.

[46] Zhang, R.F. and R.J. Urbanowicz. A scikit-learn compatible learning classifier system. in Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion. 2020.

[47] Xie, X., et al. Application of metamorphic testing to supervised classifiers. in 2009 Ninth International Conference on Quality Software. 2009. IEEE.

[48] Dwarakanath, A., et al. Identifying implementation bugs in machine learning based image classifiers using metamorphic testing. in Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2018.

[49] Segura, S., et al., *A survey on metamorphic testing.* 2016. **42**(9): p. 805-824.

[50] Segura, S., et al. Performance metamorphic testing: Motivation and challenges. in 2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER). 2017. IEEE.

[51] Gupta, N., A.P. Mathur, and M.L. Soffa. Generating test data for branch coverage. in Proceedings ASE 2000. Fifteenth IEEE International Conference on Automated Software Engineering. 2000. IEEE.

[52] Sharma, S., U. Chandra, and P.J.I.J.o.C.I.R. Jain, A literature survey on automation of test data generation for branch coverage testing using genetic algorithm. 2017. **13**(6): p. 1521-1531.

[53] Lakhotia, K., et al., *An empirical investigation into branch coverage for C programs using CUTE and AUSTIN.* 2010. **83**(12): p. 2379-2391.

[54] Bahaweres, R.B., et al. Analysis of statement branch and loop coverage in software testing with genetic algorithm. in 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI). 2017. IEEE.

[55] Hajjar, A., T. Chen, and A. von Mayrhauser. *On statistical behavior of branch coverage in testing behavioral vhdl models*. in *Proceedings IEEE International High-Level Design Validation and Test Workshop (Cat. No. PR00786)*. 2000. IEEE.

[56] Malaiya, Y.K., et al. *The relationship between test coverage and reliability*. in *Proceedings of 1994 IEEE International Symposium on Software Reliability Engineering*. 1994. IEEE.

AUTHORS

First Author – Sadia Ashraf, Masters in Software Engineering, International Islamic University Islamabad (IIUI) –

Second Author – Dr. Aamir Nadeem, Ph.D. Software Engineering, Capital University of Science and Technology (CUST).

Third Author – Dr. Salma Imtiaz, Ph.D. Software Engineering, International Islamic University Islamabad (IIUI) –

Correspondence Author – Sadia Ashraf, Masters in Software Engineering, International Islamic University Islamabad (IIUI) –