ISSN: 2994-6433

Review Article

Journal of Sensor Networks and Data Communications

Regression Test Suite Study Using Classic Statistical Methods and Machine Learning

Abhinandan H. Patil* and Sangeeta A. Patil

Educational Content Creators at 14AISS, Karnataka, India

*Corresponding Author
Abhinandan H. Patil, Senior IEEE Member, Karnataka, India.

Submitted: 2024, Jan 10; **Accepted**: 2024, Feb 13: **Published**: 2024, Feb 15

Citation: Patil, A. H., Patil, S. A. (2024). Regression Test Suite Study Using Classic Statistical Methods and Machine learning. *J Sen Net Data Comm,* 4(1), 01-05.

Abstract

This work is interdisciplinary in nature. This work tries to apply latest discoveries in Artificial Intel-ligence to classic testing methodologies. Machine Learning which is the field of Artificial Intelligence is explored in this work. The work demonstrates that provided the test team maintains the required data, Machine Learning Algorithms can aid in deciphering patterns from the test data. Patterns of interest are the relation between testers experience in the project and bugs uncovered, relations between the testers experience and the efficiency of test case with respect to code coverage and test execution time. Relation between testers experience and efficiency of test case with respect to code coverage and execution time, relation between testers experience and bugs uncovered are explored using classic statistical techniques and clustering Machine Learning Algorithms. This clustering can be of immense help in test selection, prioritization, pruning and Regression test execution time reduction.

Keywords: Clustering, Statistical Techniques, Weka, New Metric, Machine Learning, Data Science.

1. Introduction

This work assumes few things. Further the work tries to keep things as simple as possible. The idea of keeping things simple is with a reason. The final goal is to give workable solution for projects of Industrial scale. Some readers may find few of the details debatable. Few things assumed are:

• More time tester spends into a given project, their effectiveness or rather the test cases devel-oped by the given tester improves.

• While theoretically complex solutions tend to be appealing, their application to the projects of Industrial scale is far and few.

• The efficiency of the hand crafted test cases is related to time spent by tester in a given project.

• Machine Learning where used, the number of attributes or features are kept minimal to avoid running into dimensional issues and to retain the effectiveness of the solution proposed.

The book by Author Abhinandan H. Patil lays foundation on studying the relation between effec-tiveness of the test suite and number of product lines of code traced while having least number of lines in the test case and having least test execution time [1]. For the benefit of readers, the table is re-introduced.

SI No	LOCTesti	LOCProdi	Ti	Nmi
1				
2				
.				
.				
n				
Effectiveness of the total test suite = Σ Nmi Effectiveness of the test suite per test case = (Σ Nmi)/n				

Table 1: Data Maintained for Nmi.

where:
LOCTesti is Lines of Code in the test case number "i"
LOCProdi is Product Lines of code traced by the test case "i"
Ti is the test case execution time of test case "i"
Nmi = (LOCProdi) / (LOCTesti X Ti) (1)

In plain English the test case which traces maximum product lines of code with least number of lines in itself in least time is termed efficient.

The Author Abhinandan H. Patil gives hint that this can be starting point for many things incluing application of classical statistical techniques and Machine Learning on this type of data [1]. The work here starts where the book left work as an exercise for the readers. All the tables used in this work will consist of rows where each row is the per test case information. Each column is the attribute or feature of the test case.

2. Historical Data to be Maintained between Successive Execution of the Test Cases

The work assumes that where possible the test team maintains the following information:
• Total months spent by the test case developer in testing the product.
• LOCTesti, LOCProdi, Ti where each one of them retain the meaning explained in section 1.
• Number of bugs uncovered by the test case.

Texp	LOCTesti	LOCProdi	Ti	Nmi	Wbug

Table 2: Data Maintained between Successive Execution.

Where:
Texp is testers experience in months in given projects
LOCTesti, **LOCProdi, Ti, Nmi** retain the usual meaning
Wbug is appropriate weight assigned for the bugs uncovered
We introduce **Nm2i** which is simply new second metrics. Let us define the **Nm2i** as:

Nm2i = (Wbug X LOCProdi) / (LOCTesti X Ti) (2)

3. Classical Statistical Analysis

Classical Statistical analysis in expected situation would reveal peculiar pattern. The following rela-tions should be straight forward regression lines:
1) Relation between Texp and Nmi
2) Relation between Texp and Bugs uncovered
3) Relation between Texp and Nmi2

But this is very intuitive and expected behavior of the test suite. This is the precursor or preamble to the work that we will be building upon in the subsequent sections. Rather than employing these classical methods we will be exploring the Machine Learning techniques to be explained in the sub-sequent sections.

4. Clustering of Data with Machine Learning

In Clustering the required data is plotted on a two dimensional graph where each point is instance.
Associating unique number with the instances is clustering. By assigning the unique number with the instances we cluster the data.

Some clustering Algorithms lead to each instance associating with one and only one class. This is exclusive clustering.

Some clustering Algorithms lead to instances associating with more than one class. This will lead to venn diagrams with overlapping clusters.

Other Algorithms lead to hierarchical clusters called Dendrograms. Dendrograms are essentially tree structures.

Unlike in classical approach, we will be employing latest Machine Learning Algorithm to cluster the data in this section. Let us re-visit the Table 2. We have all the required information to train the computational machines at our disposal using the Machine Learning Algorithms. Since we do not have the associated class information with each test case, we leave it to computational machine for decoding. We pass 5 attributes excluding the derived attribute of Table 2 to the Machine Learning Algorithms. The output of the Algorithm will be Clustered data. These clusters are "Buckets" in standard terminology of testers. A rush through the buckets/clusters will tell which clusters/buckets are priority. The clusters could be ranked. Further these rankings will help in test selection, prioriti-zation, pruning and Regression test execution time reduction using the prioritization.

The standard Algorithms at our disposal are:
1) K Means Clustering
2) X Means Clustering
3) EM Classification
4) Cobweb Clustering
5) Hierarchical Clustering

We will not be getting into details of how these Algorithms

work or how they are implemented. These Algorithms come as prepackaged in library or tool that we will be using.

5. Weka as Machine Learning Tool

We will be using Weka as a Machine Learning Tool for our study however any other tool could be used.

The Authors prefer to use Weka for its simplicity, flexibility and capabilities.

Weka is a full-fledged tool developed using Java. The owner of the tool is University of Waikato, New Zealand. The tool comes with exceptionally well documented manual to aid the users. Weka could be invoked using:
1) Command line in turn using the shell scripts.
2) Using Graphical User Interface.
3) Using external code with Weka as library.

Weka could be extended for the specific purpose as the tool is open source. However, users seldom run into this situation. As mentioned earlier the tool is full-fledged and evolving.

6. Data Supplied to the Weka

```
@RELATION testsuite

@ATTRIBUTE testexp      REAL
@ATTRIBUTE loctesti     REAL
@ATTRIBUTE locprodi     REAL
@ATTRIBUTE ti           REAL
@ATTRIBUTE weightforbug REAL

@DATA
24,100,1000,20,100
24,150,1200,25,200
24,120,1000,22,150
24,200,2200,25,200
24,50,750,10,0
24,75,1000,15,100
24,200,2000,23,100
24,300,2800,60,400
24,400,3800,40,300
24,225,2000,35,220
24,275,2700,26,180
24,325,3000,64,300
24,350,3500,50,400
24,225,2300,40,230
24,250,2800,45,280
16,175,1600,20,100
16,120,1300,22,90
16,160,1500,24,105
16,140,1400,23,140
16,120,1200,20,100
16,180,1700,25,160
16,150,1600,26,140
16,200,1900,18,180
16,130,1500,16,120
16,100,1200,20,90
16,120,1300,22,90
16,180,1700,25,160
8,200,2800,55,180
8,250,2000,60,150
8,300,3500,70,350
8,220,2300,38,200
8,100,1200,19,100
8,140,1500,22,150
8,160,1800,32,110
8,180,2000,35,300
8,170,2000,30,280
8,400,3400,70,350
2,30,100,10,15,0
2,40,200,12,15,10
2,80,400,20,22,20
2,50,500,19,24,25
```

7. Weka Run Information

```
=== Run information ===

Scheme:       weka.clusterers.EM -I 100 -N 4 -X 10 -max -1 -ll-cv
1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100
Relation:     testsuite
Instances:    41
Attributes:   5
              testexp
              loctesti
              locprodi
              ti
              weightforbug
Test mode:    evaluate on training data

=== Clustering model (full training set) ===

EM
==

Number of clusters: 4
Number of iterations performed: 3

                Cluster
Attribute            0       1       2       3
                (0.18)  (0.22)  (0.51)   (0.1)
============================================================
=========
testexp
  mean               8      24 17.2909  2.0254
  std. dev.          0    7.71   5.066  0.7472

loctesti
  mean         243.117 284.8986 136.9517      50
  std. dev.    75.4945 61.8854 38.9257 18.6975
```

```
locprodi
  mean          2547.9978 2799.6796 1400.1295  300.5195
  std. dev.      620.8212  556.3647   325.276    158.76

ti
  mean            50.561   43.1101   21.3043   15.2439
  std. dev.      15.6316   12.7002    4.3058    4.3241

weightforbug
  mean          254.0893  280.2579  119.1436   18.9781
  std. dev.      78.9008   76.2082   41.8379    4.1106


Time taken to build model (full training data) : 0.04 seconds
=== Model and evaluation on training set ===

Clustered Instances

0      10 ( 24%)
1       9 ( 22%)
2      18 ( 44%)
3       4 ( 10%)

Log likelihood: -22.15541

8. Weka Clustered Data
@relation testsuite_clustered

@attribute Instance_number numeric
@attribute testexp numeric
@attribute loctesti numeric
@attribute locprodi numeric
@attribute ti numeric
@attribute weightforbug numeric
@attribute Cluster {cluster0,cluster1,cluster2,cluster3}

@data
0,24,100,1000,20,100,cluster2
1,24,150,1200,25,200,cluster2
2,24,120,1000,22,150,cluster2
3,24,200,2200,25,200,cluster1
4,24,50,750,10,0,cluster2
5,24,75,1000,15,100,cluster2
6,24,200,2000,23,100,cluster2
7,24,300,2800,60,400,cluster1
8,24,400,3800,40,300,cluster1
9,24,225,2000,35,220,cluster1
10,24,275,2700,26,180,cluster1
11,24,325,3000,64,300,cluster1
12,24,350,3500,50,400,cluster1
13,24,225,2300,40,230,cluster1
14,24,250,2800,45,280,cluster1
15,16,175,1600,20,100,cluster2
16,16,120,1300,22,90,cluster2
17,16,160,1500,24,105,cluster2
18,16,140,1400,23,140,cluster2
19,16,120,1200,20,100,cluster2
20,16,180,1700,25,160,cluster2
21,16,150,1600,26,140,cluster2
22,16,200,1900,18,180,cluster2
23,16,130,1500,16,120,cluster2
24,16,100,1200,20,90,cluster2
25,16,120,1300,22,90,cluster2
26,16,180,1700,25,160,cluster2
27,8,200,2800,55,180,cluster0
28,8,250,2000,60,150,cluster0
29,8,300,3500,70,350,cluster0
30,8,220,2300,38,200,cluster0
31,8,100,1200,19,100,cluster0
32,8,140,1500,22,150,cluster0
33,8,160,1800,32,110,cluster0
34,8,180,2000,35,300,cluster0
35,8,170,2000,30,280,cluster0
36,8,400,3400,70,350,cluster0
37,2,30,100,10,15,cluster3
38,2,40,200,12,15,cluster3
39,2,80,400,20,22,cluster3
40,2,50,500,19,24,cluster3
```

9. Steps to be Followed in Weka 3-9-4

1) Preprocess tab open Testsuite.arff
2) Filter→ Choose → unsupervised → Attribute → AddCluster
3) Clustertab → EM → Number of Cluster = 4 and Rest all default
4) Clustertab → start
5) Right Click on the result list → Visualize Cluster Assignments → Save results as arff

10. Result and Result Analysis

Before we start the result and result analysis few things need to be discussed. The supplied data which is the characteristic of the "Hypothetical" test suite assumes few things. The test team comprises of 4 testers with 24, 16, 8 and 2 months of experience into the project. Since the tester who has spent more time with the product tends to be well versed with the system and testing he/she will produce efficiently crafted test cases. Hypothetical data supplied to the Weka tries to emulate the real test suite as far as possible. The supplied data has taken into consideration few facts about the relation between testers experience and effectiveness of the test case {LOCTesti, LOCProdi, Ti, BugsUncoveredWeight}. We see that test cases could be clustered in the number of buckets re-quired. Further cluster0 tends to be the best cluster. Followed by the cluster 1 and 2. cluster3 con-tains the test cases crafted by novice tester. This is expected as the tester is still at the starting point of learning curve. Most of the test cases crafted by tester with 8 months experience are in cluster0. Test cases crafted by tester with 16 months is mostly in cluster2. Test cases crafted by tester with 24 months are shared between cluster1 and cluster2. While labelling the test cases on the basis of few attribute such as experience of the tester in the product testing can be controversial from few readers perspective, it is unavoidable as the result could be analyzed from

various perspectives. Management could use the data from Return on Investment (ROI) angle. These are unavoidable situation when the test result is analyzed from various perspective. This work consciously wants to avoid such controversy.

11. Conclusion

While few things are unavoidable, Data Science and Machine Learning has its own advantages. The positive side of this work is that the work could be used for test selection, test suite prioritization, pruning and Regression test suite execution time reduction among many other things [2-4].

Acknowledgement

The Authors of this work are indebted to Weka team for the wonderful tool and library. Authors acknowledge the open source operating system Kubuntu 19.10 for ease of use and capabilities. Many thanks to OpenJDK. Many thanks to the LibreOffice team for the wonderful documentation software.

References

1. Patil, A. H. (2019). *Design and implementation of combinatorial testing based test suites for operat-ing systems used for internet of things*. Lulu. com.
2. Witten, I. H., Frank, E., Hall, M. A., Pal, C. J., & Data, M. (2005, June). Practical machine learning tools and techniques. In *Data mining* (Vol. 2, No. 4, pp. 403-413). Amsterdam, The Netherlands: Elsevier.
3. Witten Ian H., Frank Eibe, Hall Mark A., and Pal Christopher. (2016). Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques. *Morgan Kaufmann Publishers.*
4. University of Waikato. Weka Manual for version 3-9-3.