CAP6673 – Data Mining and Machine Learning

Assignment 2: Classification Using Decision Trees

Written by:

Christopher Foley
Z15092976

Academic Year: 2016-2017

s

Table of Contents

1 Introduction

The intent of this paper is to review and hopefully replicate some of the work of T.M. Khoshgoftaar and E.B. Allen, in their paper *Classification of Fault-Prone Software Modules, Prior Probabilities, Costs and Model Evaluations.*[1]

The data was prepared using the directions in the class assignment #1 as Data Set II. Modules with 2 or more faults were marked as fault prone (fp) modules with less than 2 were marked as not fault prone (nfp). Files were prepared using LibreOffice Calc and a macro was used to populate the FaultProne field as instructed in Assignment 1 (=if(A$9<2,'nfp','fp')) the ARFF file was then processed using the WEKA analysis tool.

The goal of this assignment is to classify modules as fault prone (fp) or not fault prone (nfp) based upon a decision tree.

Data

The data provided appears to be a subset of the data created by a large command, control and communications system analyzed earlier [i] Data was input to the WEKA tool and the NFAULTS field excluded from analysis. Using the author provided references the following product metrics were studied:

1 T.M. Khoshgoftaar and E.B. Allen, *Class ification of Fault-Prone Software Modules: Prior Probabilities, Costs and Model Evaluations,* copy provided by author T.M. Khoshgoftaar as part of class notes CAP6674: Machine Learning and Data Mining, Florida Atlantic University, Spring 2017.

s

Attribute	Description
NUMORS	Number of Unique operators
NUMANDS	Number of Unique operands
TOTOTORS	Total number of operators
TOTOPANDS	Total number of operands
VG	McCabe Complexity Complex
NLOGIC	Number of logical ooperators
LOC	Number of lines of code
ELOC	Executable lines of code
Fp/nfp	Fault Prone / Not Fault Prone

A FIT data set containing 184 instances was used with the WEKA tool to train the model and create initial data. Two basic models were built, a linear regression model and a Decision Stump model. After creating models with the FIT data, TEST data was then used and the results compared.

2 Initial Tree

Fit Data Set

The first goal was to "predict a class (fp, nfp) using J48 (C4.5), a decision tree based classification algorithm."[2] The first test was with a pruned tree, branches that do not affect the output are removed, and the default confidence level. The

The WEKA tool created the following classifier data:

```
=== Classifier model (full training set) ===


J48 pruned tree
------------------
```

2 T.M. Khoshgoftaar , Assignment 2 based on "CAP 6673: Data Mining and Machine Learning",

s

```
NUMUANDS <= 115
|   ELOC <= 50: nfp (106.0/2.0)
|   ELOC > 50
|   |   ELOC <= 105
|   |   |   NLOGIC <= 0: nfp (12.0)
|   |   |   NLOGIC > 0
|   |   |   |   ELOC <= 54: fp (6.0/1.0)
|   |   |   |   ELOC > 54: nfp (5.0)
|   |   ELOC > 105: fp (5.0)
NUMUANDS > 115
|   VG <= 29: fp (16.0)
|   VG > 29
|   |   LOC <= 1158: nfp (10.0/1.0)
|   |   LOC > 1158: fp (28.0/2.0)

Number of Leaves  :     8

Size of the tree :  15
```

The tree contained two observations of note. First when there were more than 115 unique operands the lines of code (versus the Executable lines of code) were significant. Second, but not surprising, that modules with smaller numbers of lines of code are likely to be not fault prone.

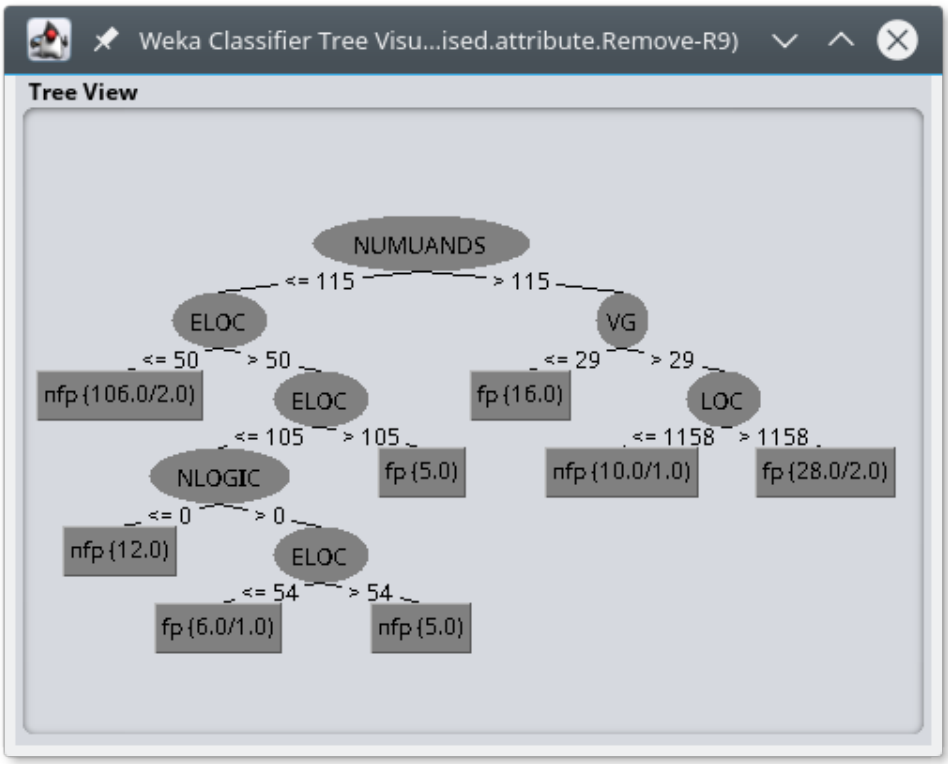
s

The confusion showed nearly equal numbers of modules misclassified (fp classified as nfp and nfp classified as fp).

```
=== Confusion Matrix ===

   a   b   <-- classified as
 121  12 |   a = nfp
  11  44 |   b = fp
```

The initial FIT data produced the following tree:



Test Data Set

Similar results occurred when the test data set was used with the training data. Classifier data was given as:

```
=== Classifier model (full training set) ===
```

s

```
InputMappedClassifier:

J48 pruned tree
------------------

NUMUANDS <= 115
|   ELOC <= 50: nfp (106.0/2.0)
|   ELOC > 50
|   |   ELOC <= 105
|   |   |   NLOGIC <= 0: nfp (12.0)
|   |   |   NLOGIC > 0
|   |   |   |   ELOC <= 54: fp (6.0/1.0)
|   |   |   |   ELOC > 54: nfp (5.0)
|   |   ELOC > 105: fp (5.0)
NUMUANDS > 115
|   VG <= 29: fp (16.0)
|   VG > 29
|   |   LOC <= 1158: nfp (10.0/1.0)
|   |   LOC > 1158: fp (28.0/2.0)

Number of Leaves  :     8

Size of the tree :  15
```

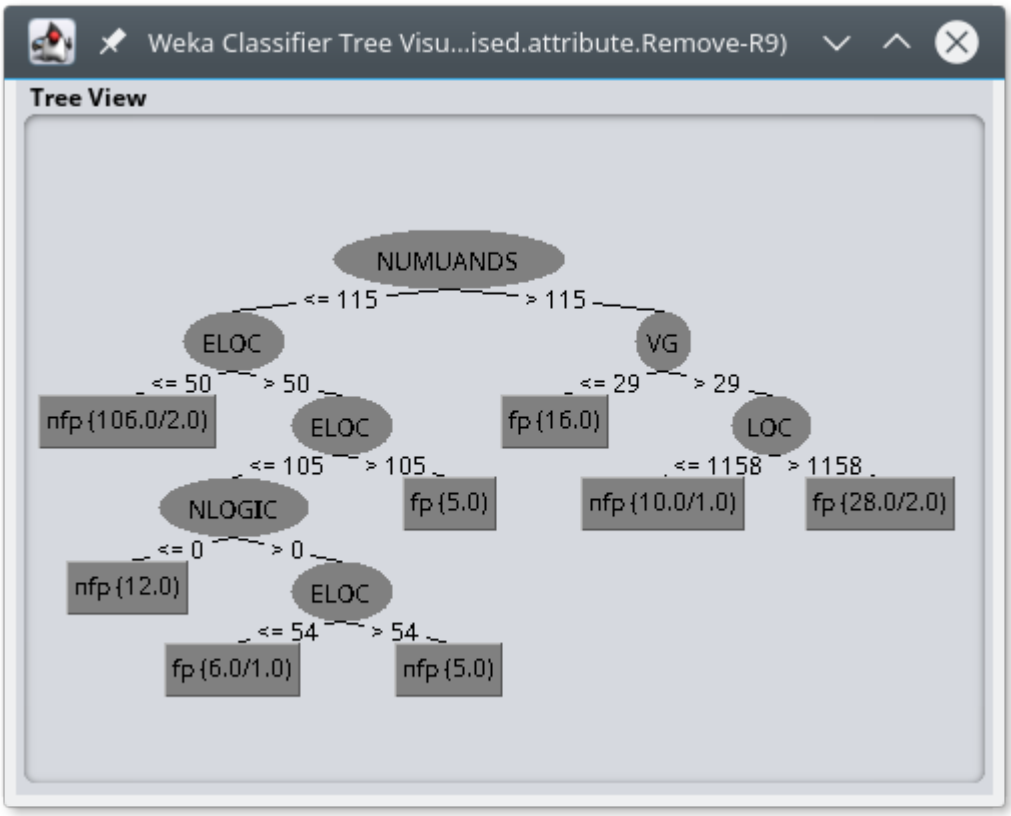
Surprisingly the TEST data set classified more modules as NFP that were FP. The confusion matrix was as follows:

s

```
=== Confusion Matrix ===

   a   b   <-- classified as
  61   5 |   a = nfp
   9  19 |   b = fp
```

The TEST data used the same decision tree as the training data:



3 Unpruned Tree

As expected the unpruned tree contained more leaves and was larger, however not by much. The unpruned tree only had one more leaf and the size was larger. What was surprising, from a software development orientation, was that the pruned node contained the number of operators and not the

s

lines of code which contains comments..

FIT Data set

The FIT data set produced the following tree:

=== Classifier model (full training set) ===

J48 unpruned tree

NUMUANDS <= 115

| ELOC <= 50: nfp (106.0/2.0)

| ELOC > 50

| | ELOC <= 105

| | | NLOGIC <= 0: nfp (12.0)

| | | NLOGIC > 0

| | | | ELOC <= 54: fp (6.0/1.0)

| | | | ELOC > 54: nfp (5.0)

| | ELOC > 105: fp (5.0)

NUMUANDS > 115

| NUMUORS <= 42

| | VG <= 29: fp (16.0)

| | VG > 29

| | | LOC <= 1158: nfp (10.0/1.0)

| | | LOC > 1158: fp (10.0/2.0)

| NUMUORS > 42: fp (18.0)

s

Number of Leaves : 9

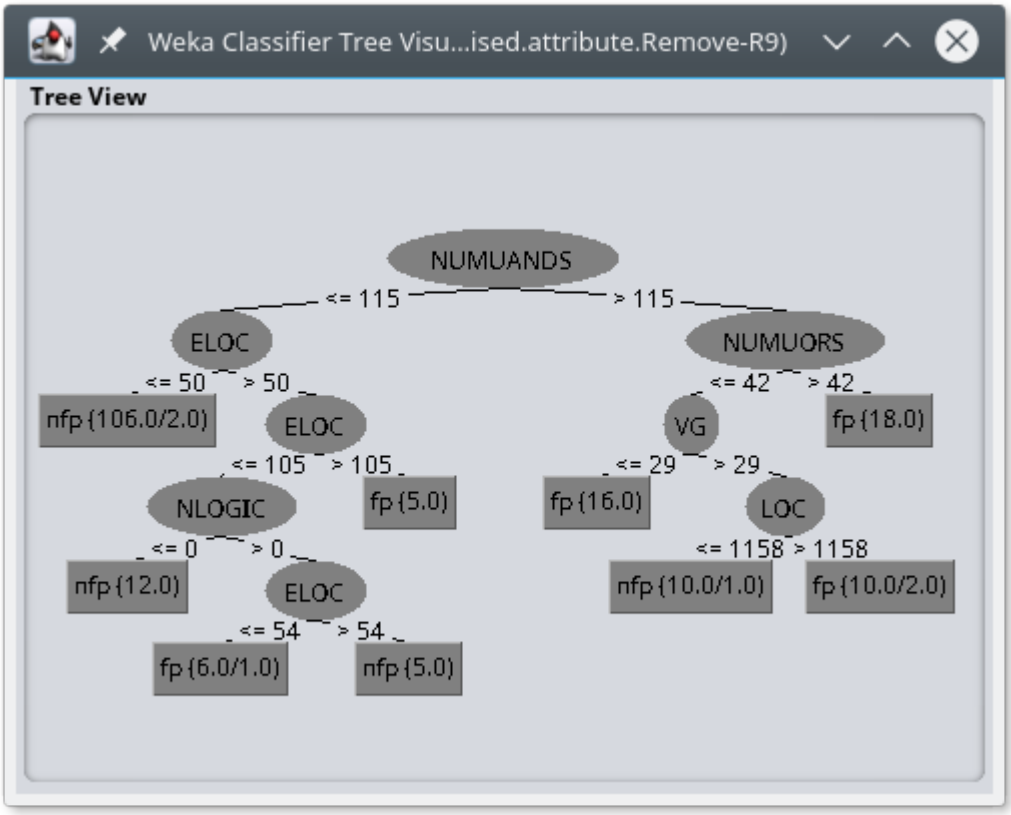
Size of the tree : 17

The confusion matrix showed similar numbers of misclassifications.

=== Confusion Matrix ===

```
   a   b   <-- classified as
 121  12 |   a = nfp
  11  44 |   b = fp
```

The graphical tree appeared more balanced, though.



s

TEST Data Set

The TEST data set contained the same results.

=== Classifier model (full training set) ===

InputMappedClassifier:

J48 unpruned tree

NUMUANDS <= 115

| ELOC <= 50: nfp (106.0/2.0)

| ELOC > 50

| | ELOC <= 105

| | | NLOGIC <= 0: nfp (12.0)

| | | NLOGIC > 0

| | | | ELOC <= 54: fp (6.0/1.0)

| | | | ELOC > 54: nfp (5.0)

| | ELOC > 105: fp (5.0)

NUMUANDS > 115

| NUMUORS <= 42

| | VG <= 29: fp (16.0)

| | VG > 29

| | | LOC <= 1158: nfp (10.0/1.0)

| | | LOC > 1158: fp (10.0/2.0)

| NUMUORS > 42: fp (18.0)

s

Number of Leaves : 9

Size of the tree : 17

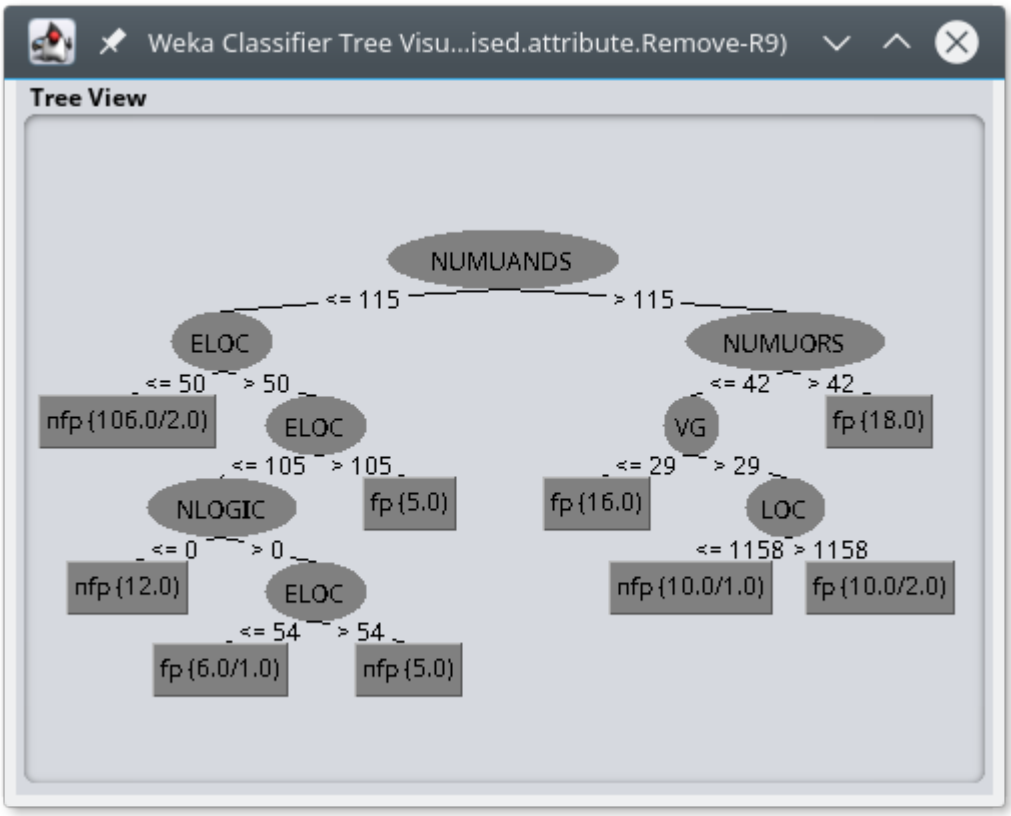
Although the Confusion Matrix showed a higher number of Type-II misclassifications.

=== Confusion Matrix ===

```
 a  b   <-- classified as
61  5 |  a = nfp
 8 20 |  b = fp
```

The graphic tree was as expected.

s



4 Confidence Factor

Another part of the examination was to vary the confidence (accuracy) factor from 0.25 (default)) to 0.01.

FIT data set

When confidence is reduced, the pruning is more significant. The pruned tree reduced the number of leaves from 8 to 8 and the size from 15 to 7. The tree was much closer to the models produced in assignment 1. Differing from from Assignment 1, which determined the number of logical operators determined the classifications, the reduced accuracy indicated that the number of operands and the McCabe complexity quotient lead to the determination of fault prone.

=== Classifier model (full training set) ===

J48 pruned tree

s

NUMUANDS <= 115: nfp (134.0/12.0)

NUMUANDS > 115

| VG <= 29: fp (16.0)

| VG > 29

| | LOC <= 1158: nfp (10.0/1.0)

| | LOC > 1158: fp (28.0/2.0)

Number of Leaves : 4

Size of the tree : 7

However the simplicity does not lead to accuracy. Reducing accuracy, leads to a doubling of Type-II errors which are more costly in terms of a product development prospective.

=== Confusion Matrix ===

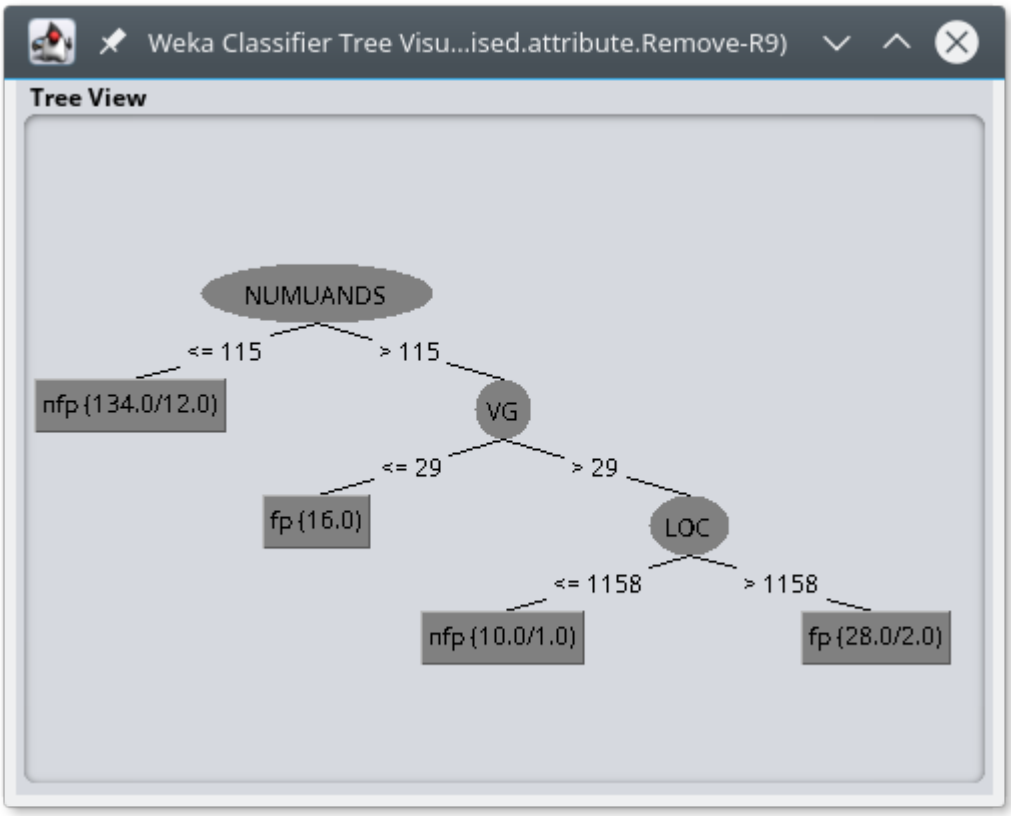
a b <-- classified as

120 13 | a = nfp

15 40 | b = fp

The graphic is significantly more simplified than other models.

s



TEST data set

The reduced error rates are more pronounced with the TEST data set.

=== Classifier model (full training set) ===

InputMappedClassifier:

J48 pruned tree

----

s

NUMUANDS <= 115: nfp (134.0/12.0)

NUMUANDS > 115

| VG <= 29: fp (16.0)

| VG > 29

| | LOC <= 1158: nfp (10.0/1.0)

| | LOC > 1158: fp (28.0/2.0)

Number of Leaves : 4

Size of the tree : 7

The number of Type-II misclassifications is nearly triple that of Type-I.

=== Confusion Matrix ===

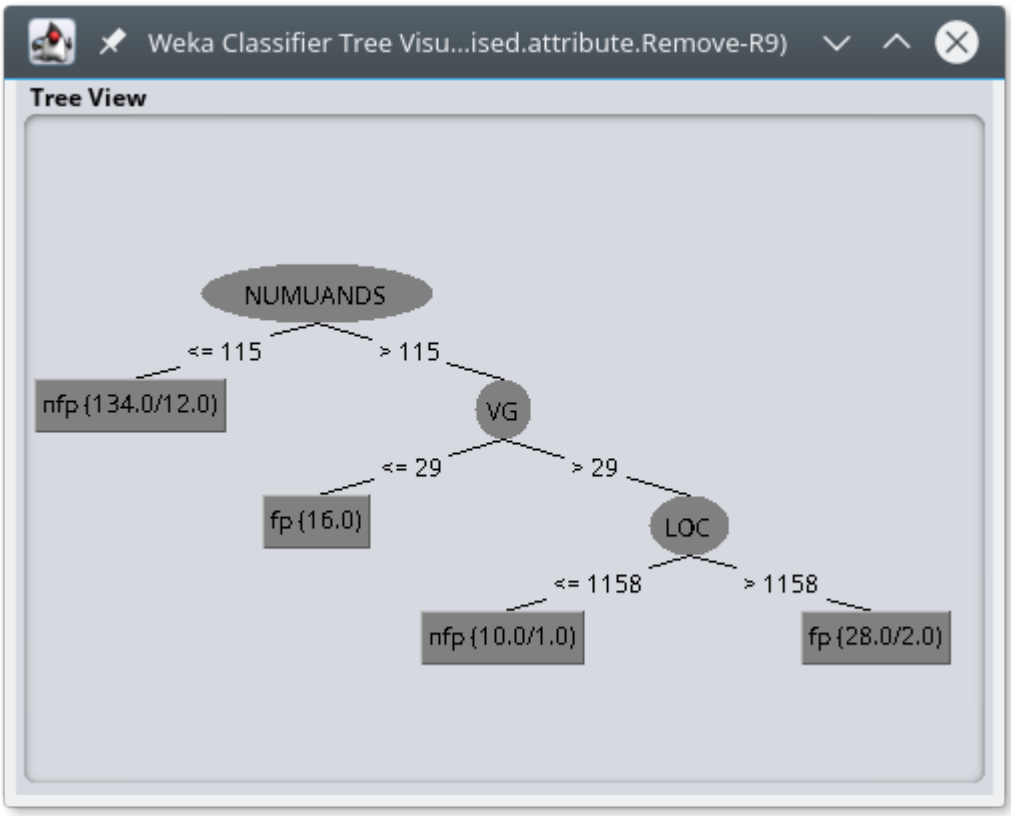
a b <-- classified as

62 4 | a = nfp

11 17 | b = fp

The Graphical tree is as expected.

s



5 Cost Sensitivity

In the experiments, to date, Type-I and Type-II misclassifications were treated as equal. However, it is obvious that a False Positive is rarely equal in impact to a false negative. In software engineering, classifying a module as fault prone that contains no faults has less of an impact than a module with errors classified as not fault prone.

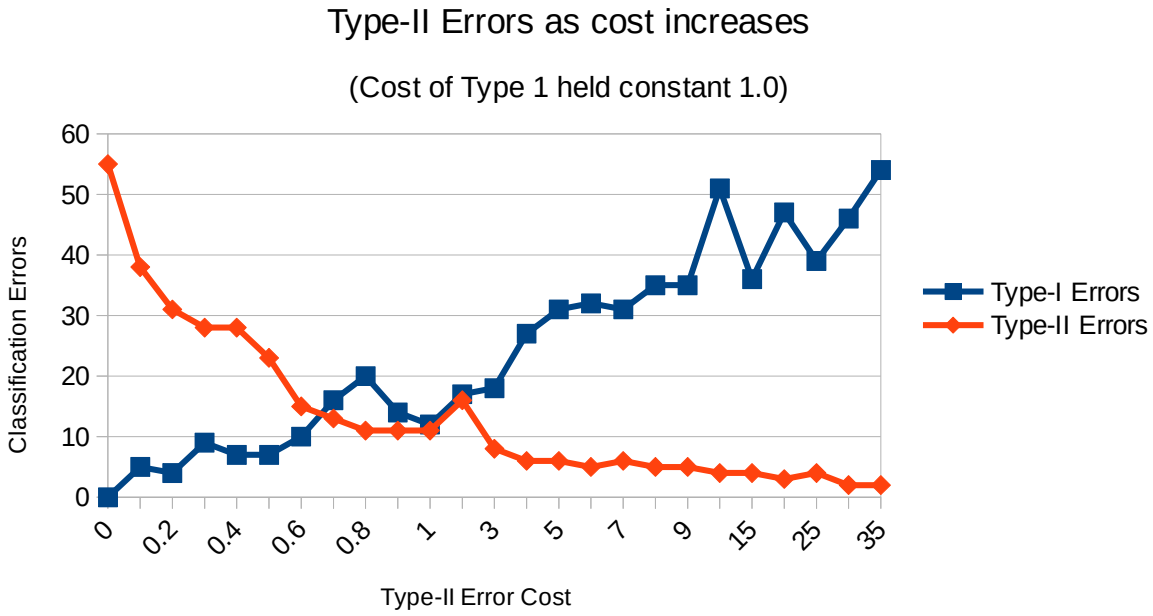
We then used the cost sensitive classifier combined with J48, and determine the optimal cost ratio (set cost of a type I error to 1 and vary the cost of the Type II error), using 10-fold cross validation on the fit data set. The Cost of a Type-I error was fixed at 1 and the Type-II cost varied.

The following table and chart indicates the results. Of note is that as the cost of Type-II errors rise above the cost of Type-I errors the cost 2 errors reduce and the Type-I errors increase. The error rates appear to be equal when the cost of a Type-II error is around 0.6 more than the cost of a Type-I error.

s

Confidence factor	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	2
Type-I	0	5	4	9	7	7	10	16	20	14	12	17
Type-II	55	38	31	28	28	23	15	13	11	11	11	6

Confidence factor	3	4	5	6	7	8	9	10	15	20	25	30
Type-I	18	27	31	32	31	35	35	51	36	47	39	46
Type-II	8	6	6	5	6	5	5	4	4	3	4	2



6 Conclusions

Analysis of the data indicates an, not surprising, axiom of Computer Science. Errors that are not caught in development are costly. What is surprising is that errors may be reasonably. Predicted using the number of operands and executable lines of code.

The cost analysis showed clearly that as the cost of false negative (Type-II) errors increases, more

s

errors are found. The break even point appears to be when the Type-II errors are between 1 and 2 times more costly than type 1 errors.

More research will attempt to predict module failure before release.

..

..

…

..

i T.M. Khoshgoftaar and E.B. Allen, *Modeling Software Quality: The* Software *Measurement Analysis and Reliability Toolkit*, copy provided by author T.M. Khoshgoftaar as part of class notes CAP6674: Machine Learning and Data Mining, Florida Atlantic University, Spring 2017.