CAP6673 – Data Mining and Machine Learning

# Using meta learning schemes with a strong and a weak learner

Written by:

**Christopher Foley**
**Z15092976**

Academic Year: 2016-2017

## Table of Contents

## I. Problem Specification

The problem is described on the class web site as follows:

> Using the same methods as above (10-fold cross-validation on the fit data), determine the preferred (optimal cost ratio) model, and evaluate the models using the test dataset for:
>
> 1. Cost sensitive classifier combined with bagging and J48
> 2. Cost sensitive classifier combined with bagging and Decision Stump
> 3. Cost sensitive classifier combined with boosting (AdaBoostM1)and J48
> 4. Cost sensitive classifier combined with boosting (AdaBoostM1) and Decision Stump
>
> Use the default settings for the meta learners (bagging, boosting, ) and the learner (J48, decision stump) but vary the cost ratio in the same way as in Part 4 of Assignment II. Provide the command lines you used.
>
> How do the results of each classifier compare to the cost-sensitive tree obtained in Part 4 of Assignment II? Comment.
>
> Now, set the number of iterations of each meta learner to 25 and repeat the experiments. Don't forget to analyze the results.

## II. The Data

The data is two .arff files created for the class based on prior work by the professor. Using the methodology demonstrated in class by Matthew Herland on 1/25/17 the data files were altered from those used in Assignment II to remove the field NUMFAULTS, leaving the FP, NFP field intact. An excerpt of the data files is as follows:

```
% CCCS - FIT  data set for Data mining classes

% classification assignment

% Author: Christopher Foley

% email: cfoley3@fau.edu

% 09 attributes


@relation FIT
```

```
@attribute NUMUORS real
@attribute NUMUANDS real
@attribute TOTOTORS real
@attribute TOTOPANDS real
@attribute VG real
@attribute NLOGIC real
@attribute LOC real
@attribute ELOC real
@attribute FAULTPRONE {nfp, fp}

@data
22,85,203,174,9,0,362,40,nfp
21,87,186,165,5,0,379,32,nfp
30,107,405,306,25,0,756,99,nfp
```

This was done to assist in reducing the time needed to conduct the tests.

## III. The Tests

For this series of tests the command line was used with a bash script running under OpenMandriva LX3 (Linux). A script similar to the following was used:

```
# /bin/bash -x
# command to run  weka

for iter in 10 25 ; do
 for cost in 0.25 0.5 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 ; do
    java -classpath weka-3-8-1/weka.jar
weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 ${cost}; 1.0
0.0]" -S 1 -W weka.classifiers.meta.Bagging  -t
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/FIT3.arff -- -P 100 -S 1 -num-
slots 1 -I ${iter}  -W weka.classifiers.trees.J48  -- -C 0.25 -M 2 &>>
CostSensitive_Bagging_J48.txt


   java -classpath weka-3-8-1/weka.jar
```

```
weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 ${cost}; 1.0
0.0]" -S 1 -W weka.classifiers.meta.Bagging  -t
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/FIT3.arff  -T
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/TEST3.arff -- -P 100 -S 1 -num-
slots 1 -I ${iter} -W weka.classifiers.trees.J48 -- -C 0.25 -M 2 &>>
CostSensitive_Bagging_J48.txt


done

done

exit 0
```

Use of the script versus the GUI interface reduced the time to run the tests from an estimated 8 hours to 10 minutes, allowing more time for analysis and correction.  The full script is addached as Appendix A (page 11).  A variant of the script in appendix A was used which had minimal differences  that were added to write blank lines, script debug messages and to enhance readability of the text output.

## IV. Analysis

### Procedure for Analysis

The script shown in Appendix A (page 11) was used to generate 5 text based data files.    Each data file was examined and the false positive rate was determined from the output (FP Rate).  The FP Rate was transcribed into a spreadsheet program (LibreOffice Calc) and the rates mapped.  The Fit data, was used to train the model and the Test data set was used to determine the actual results.

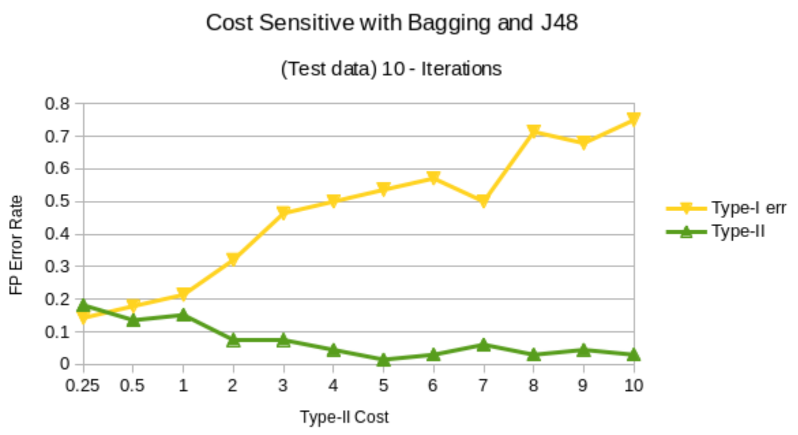
The comparison between the Type-I and Type-II data for the test data presented in this section .  The Fit data comparison  without analysis is shown in Appendix B (page 13).has been excerpted for readability.  The actual data set compared 12 cost levels for Fit and Test data for 10 and 25 iterations, thus creating 380 data points.  The data surrounding the optimal FP Rate cost is displayed in the charts and the Test data is displayed following the chart.  For comparison the FP rates from the previous assignment are shown in Appendix C (page 17).

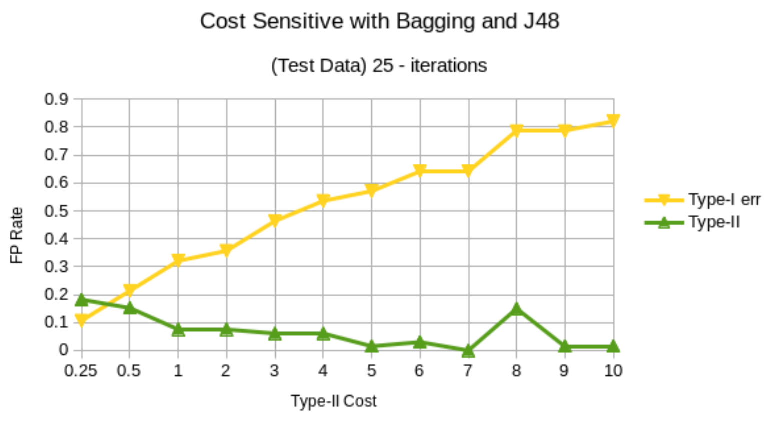
### Cost sensitive classifier combined with bagging and J48

Review of the data indicates that when the cost of Type-II errors is between 0.25 and .5 of the cost of Type-I errors the False Positive rates are nearly equal.  This is indicated in the following table and graphs.  Further research indicates that the optimal cost for Type-II is 0.36.

| **Type-II cost** | 10 – Iterations | | | | 25 – Iterations | | | |
|---|---|---|---|---|---|---|---|---|
| | Fit | | Test | | Fit | | Test | |
| | Type-I | Type-II | Type-I | Type-II | Type-I | Type-II | Type-I | Type-II |
| **0.25** | **0.109** | **0.158** | **0.143** | **0.182** | **0.073** | **0.180** | **0.107** | **0.182** |
| **0.5** | **0.182** | **0.143** | **0.179** | **0.136** | **0.145** | **0.135** | **0.214** | **0.152** |
| 1.0 | 0.200 | 0.083 | 0.214 | 0.152 | 0.200 | 0.090 | 0.321 | 0.076 |
| 2.0 | 0.291 | 0.053 | 0.321 | 0.076 | 0.309 | 0.068 | 0.357 | 0.076 |
| 3.0 | 0.40 | 0.06 | 0.464 | 0.076 | 0.418 | 0.068 | 0.464 | 0.461 |
| 4.0 | 0.491 | 0.045 | 0.500 | 0.045 | 0.545 | 0.053 | 0.536 | 0.061 |
| 5.0 | 0.545 | 0.038 | 0.536 | 0.015 | 0.564 | 0.030 | 0.571 | 0.015 |

*Table 1: Cost Sensitive Classifier combined with Bagging and J48 meta learners*



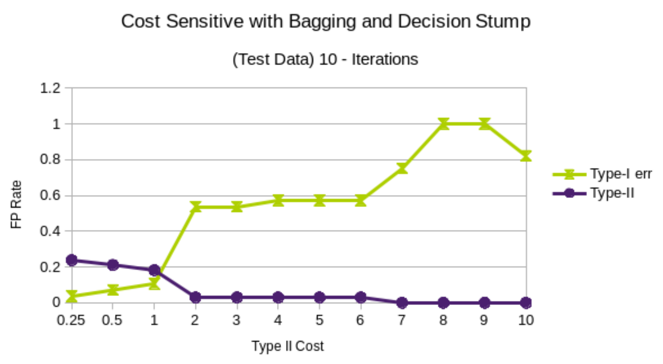


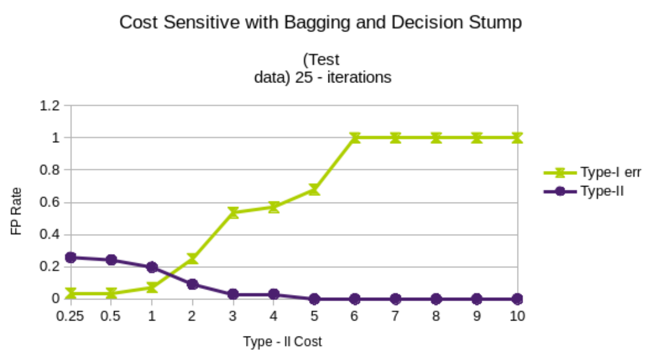
## Cost sensitive classifier combined with bagging and Decision Stump

When the cost sensitive classifier is combined with bagging and a decsision stump, the False Positive rates are nearly equal when the costs are equal.

| **Type-II cost** | 10 – Iterations | | | | 25 – Iterations | | | |
|---|---|---|---|---|---|---|---|---|
| | Fit | | Test | | Fit | | Test | |
| | Type-I | Type-II | Type-I | Type-II | Type-I | Type-II | Type-I | Type-II |
| 0.25 | 0.055 | 0.226 | 0.036 | 0.238 | 0.055 | 0.218 | 0.036 | 0.258 |
| 0.5 | 0.091 | 0.211 | 0.071 | 0.212 | 0.109 | 0.211 | 0.036 | 0.242 |
| **1.0** | **0.145** | **0.165** | **0.107** | **0.182** | **0.164** | **0.165** | **0.071** | **0.197** |
| **2.0** | **0.382** | **0.083** | **0.536** | **0.030** | **0.364** | **0.083** | **0.250** | **0.091** |
| 3.0 | 0.473 | 0.060 | 0.536 | 0.030 | 0.564 | 0.023 | 0.536 | 0.030 |
| 4.0 | 0.709 | 0.023 | 0.571 | 0.030 | 0.709 | 0.015 | 0.571 | 0.030 |

*Table 2: Cost Sensitive Classifier combined with Bagging and Decision Stump meta learners*



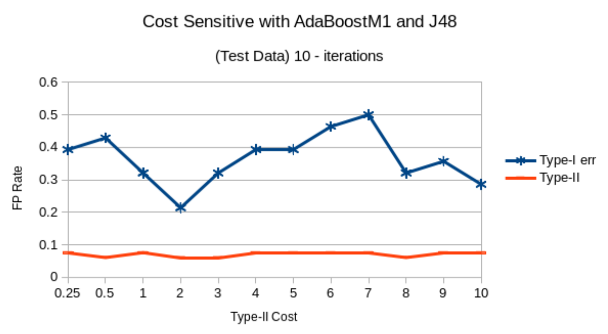


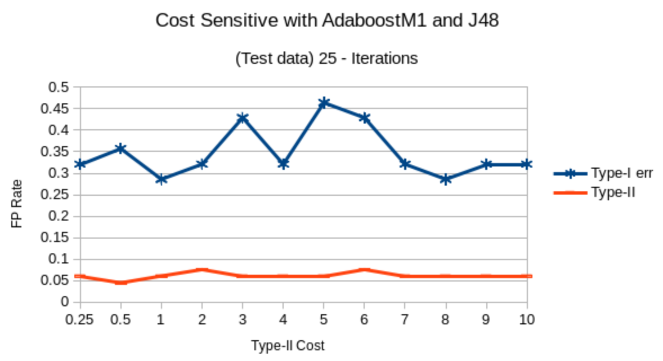
## Cost sensitive classifier combined with boosting (AdaBoostM1) and J48

As can be easily seen the cost sensitive classifier when combined with boosting and J48 decision tree does not produce an optimal point where the error rates are equal. However, as indicated by the graphs the Type-II FP rate is relatively stable. The optimally chosen solution was selected based on the cost where the Type-I and Type-II are reasonably close. In this case a Type-II cost that was double that of type I seemed to be closest. This reflects experience in the world where a undiscovered error can be more costly to detect and fix.

| **Type-II cost** | 10 – Iterations | | | | 25 – Iterations | | | |
|---|---|---|---|---|---|---|---|---|
| | Fit | | Test | | Fit | | Test | |
| | Type-I | Type-II | Type-I | Type-II | Type-I | Type-II | Type-I | Type-II |
| 0.25 | 0.273 | 0.105 | 0.393 | 0.076 | 0.182 | 0.090 | 0.321 | 0.061 |
| 0.5 | 0.182 | 0.120 | 0.429 | 0.061 | 0.182 | 0.113 | .0357 | 0.045 |
| 1.0 | 0.291 | 0.098 | 0.321 | 0.076 | 0.236 | 0.083 | 0.286 | 0.061 |
| **2.0** | **0.200** | **0.105** | **0.214** | **0.061** | **0.182** | **0.083** | **0.321** | **0.076** |
| 3.0 | 0.236 | 0.090 | 0.321 | 0.061 | 0.200 | 0.083 | 0.429 | 0.061 |
| 4.0 | 0.236 | 0.105 | 0.393 | 0.076 | 0.273 | 0.075 | 0.321 | 0.061 |

*Table 3: Cost Sensitive Classifier combined with Boosting (AdaboostM1) and J48 meta learners*





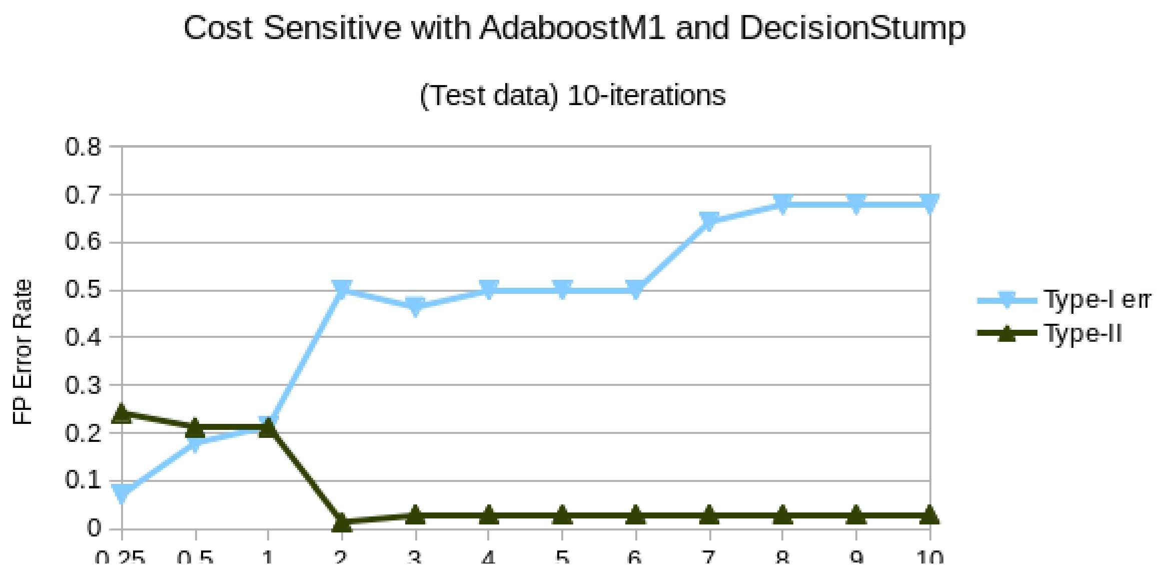
Reviewing the results of the prior exercise (HW2) in Appendix C (page 17 ), we see that the addition of the AdaboostM1 meta-learner has caused the data to stabilize but there is no obvious location where the FP rates cross. Additionally of interest is the size of the tree. In HW2 the tree was between 2-3 Leaves and a size of 3-5, EXCEPT where the FP rates converged. At that point the number of leaves was 8 and the Size if the tree is 15. When the AdaboostM1 and J48 meta-learners are added to cost sensitive classifier, the decision trees average 23 leaves and a size of 10.

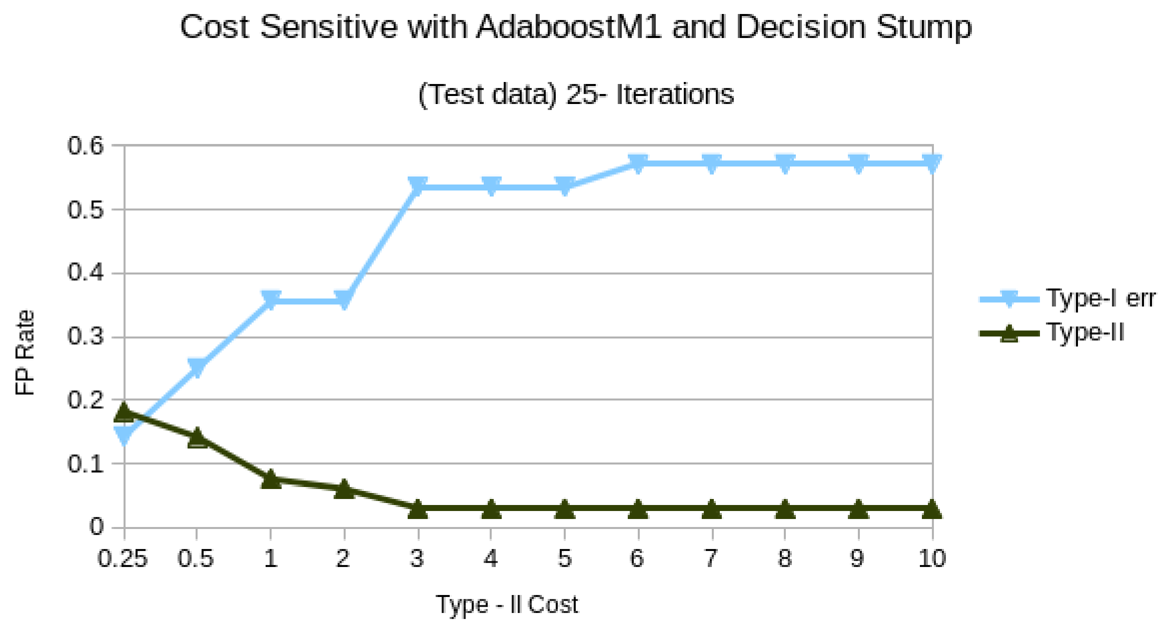
## Cost sensitive classifier combined with boosting (AdaBoostM1) and Decision Stump

A Cost sensitive classifier combined with boosting and a Decision stump did provide convergence of the FP Rates. Specifically, after 10 iterations the model converged at 1 and when the number of iterations increased the iterations converged closer to 0.25. Both Fit and Test data sets exhibited the same characteristics.

| **Type-II cost** | 10 – Iterations | | | | 25 – Iterations | | | |
|---|---|---|---|---|---|---|---|---|
| | Fit | | Test | | Fit | | Test | |
| | Type-I | Type-II | Type-I | Type-II | Type-I | Type-II | Type-I | Type-II |
| 0.25 | 0.109 | 0.211 | 0.071 | 0.242 | 0.127 | 0.188 | **0.143** | **0.182** |
| **0.5** | **0.164** | **0.173** | **0.179** | **0.212** | **0.182** | **0.143** | **0.250** | **0.142** |
| **1.0** | **0.218** | **0.158** | **0.214** | **0.212** | **0.273** | **0.133** | 0.357 | 0.076 |
| 2.0 | 0.455 | 0.083 | 0.500 | 0.014 | 0.364 | 0.060 | 0.357 | 0.061 |
| 3.0 | 0.418 | 0.083 | 0.464 | 0.030 | 0.364 | 0.068 | 0.536 | 0.030 |
| 4.0 | 0.491 | 0.053 | 0.500 | 0.030 | 0.418 | 0.060 | 0.536 | 0.030 |

*Table 4: Cost Sensitive Classifier combined with Boosting (AdaboostM1) and Decision Stump meta learners*





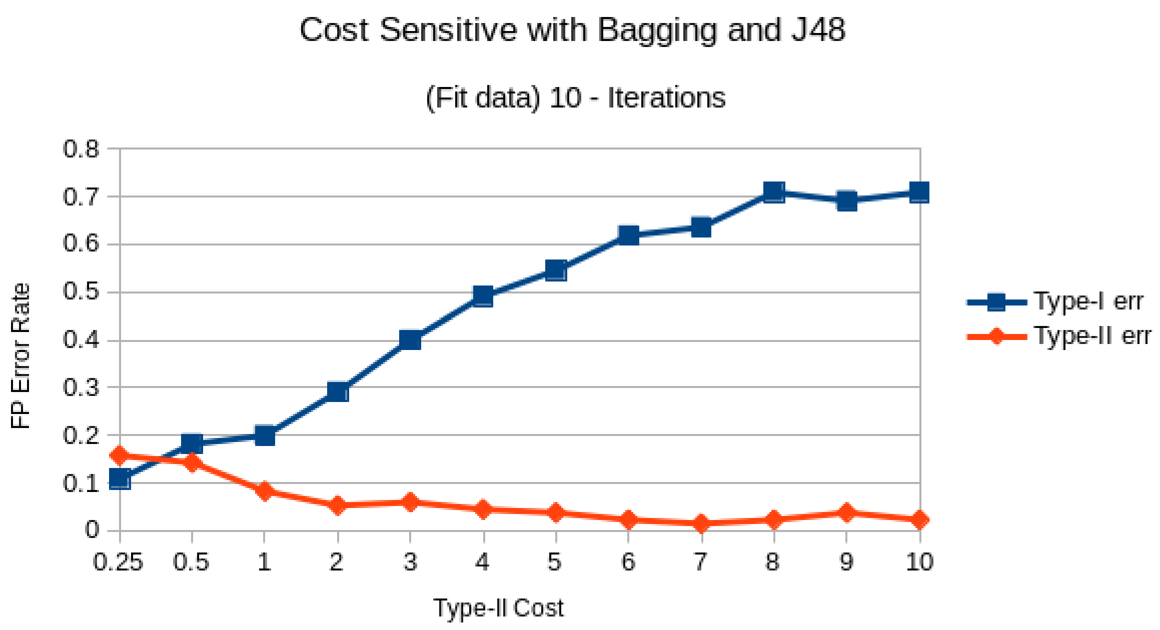
## V. Appendix A – Linux Script used

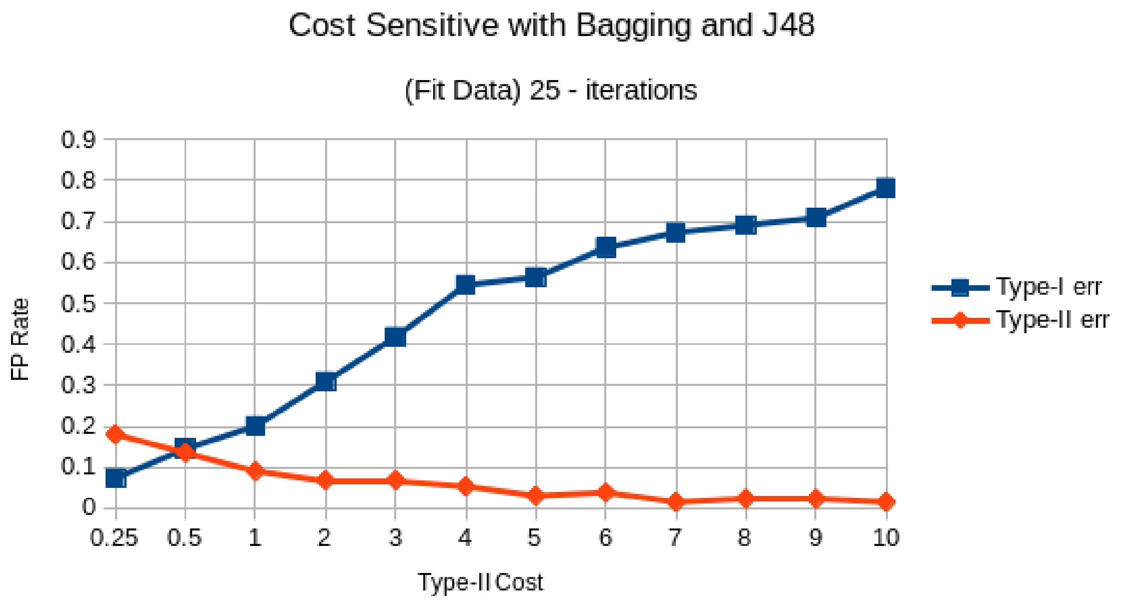
```
# /bin/bash -x

# command to run  weka


for iter in 10 25 ; do


 for cost in 0.25 0.5 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 ; do

   touch CostSensitive_Bagging_J48.txt

   touch CostSensitive_Bagging_DecisionStump.txt

   touch CostSensitive_AdaBoostM1_J48.txt

   touch CostSensitive_AdaBoost_DecisionStump.txt


    printf "\n*** FIT ***\n" >> CostSensitive_Bagging_J48.txt

    java -classpath weka-3-8-1/weka.jar  weka.classifiers.meta.CostSensitiveClassifier
-cost-matrix "[0.0 ${cost}; 1.0 0.0]" -S 1 -W weka.classifiers.meta.Bagging  -t
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/FIT3.arff -- -P 100 -S 1 -num-slots 1 -I $
{iter}  -W weka.classifiers.trees.J48  -- -C 0.25 -M 2 &>> CostSensitive_Bagging_J48.txt


    printf "\n **** TEST ****\n" >> CostSensitive_Bagging_J48.txt

   java -classpath weka-3-8-1/weka.jar  weka.classifiers.meta.CostSensitiveClassifier -cost-
matrix "[0.0 ${cost}; 1.0 0.0]" -S 1 -W weka.classifiers.meta.Bagging  -t
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/FIT3.arff  -T
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/TEST3.arff -- -P 100 -S 1 -num-slots 1 -I $
{iter} -W weka.classifiers.trees.J48 -- -C 0.25 -M 2 &>> CostSensitive_Bagging_J48.txt


   printf "\n*** FIT ***\n" >>  CostSensitive_Bagging_DecisionStump.txt

   java -classpath weka-3-8-1/weka.jar  weka.classifiers.meta.CostSensitiveClassifier -cost-
matrix "[0.0 ${cost}; 1.0 0.0]" -S 1 -W weka.classifiers.meta.Bagging  -t
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/FIT3.arff -- -P 100 -S 1 -num-slots 1 -I $
{iter} -W weka.classifiers.trees.DecisionStump &>> CostSensitive_Bagging_DecisionStump.txt


   printf "\n **** TEST ****\n" >> CostSensitive_Bagging_DecisionStump.txt

   java -classpath weka-3-8-1/weka.jar  weka.classifiers.meta.CostSensitiveClassifier -cost-
matrix "[0.0 ${cost}; 1.0 0.0]" -S 1 -W weka.classifiers.meta.Bagging  -t
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/FIT3.arff  -T
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/TEST3.arff -- -P 100 -S 1 -num-slots 1 -I $
{iter} -W weka.classifiers.trees.DecisionStump &>> CostSensitive_Bagging_DecisionStump.txt

touch CostSensitive_Bagging_J48.txt


   printf "\n*** FIT ***\n"  >>  CostSensitive_AdaBoostM1_J48.txt
   java -classpath weka-3-8-1/weka.jar weka.classifiers.meta.CostSensitiveClassifier -cost-
matrix "[0.0  ${cost}; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -t
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/FIT3.arff  -- -P 100 -S 1 -I ${iter} -W
weka.classifiers.trees.J48 -- -C 0.25 -M 2 &>> CostSensitive_AdaBoostM1_J48.txt


    printf "\n **** TEST ****\n" >>  CostSensitive_AdaBoostM1_J48.txt

       java -classpath weka-3-8-1/weka.jar weka.classifiers.meta.CostSensitiveClassifier
-cost-matrix "[0.0 ${cost}; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -t
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/FIT3.arff  -T
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/TEST3.arff -- -P 100 -S 1 -I ${iter} -W
weka.classifiers.trees.J48 -- -C 0.25 -M 2  &>> CostSensitive_AdaBoostM1_J48.txt


   printf "\n*** FIT ***\n"  >>  CostSensitive_AdaBoost_DecisionStump.txt

   java -classpath weka-3-8-1/weka.jar weka.classifiers.meta.CostSensitiveClassifier -cost-
matrix "[0.0  ${cost}; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -t
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/FIT3.arff  -- -P 100 -S 1 -I ${iter} -W
weka.classifiers.trees.DecisionStump &>> CostSensitive_AdaBoost_DecisionStump.txt


    printf "\n **** TEST ****\n" >>  CostSensitive_AdaBoost_DecisionStump.txt

    java -classpath weka-3-8-1/weka.jar weka.classifiers.meta.CostSensitiveClassifier -cost-
matrix "[0.0 ${cost}; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -t
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/FIT3.arff  -T
Dropbox/MSCS/CAP6673MachineLearningAndDataMining/TEST3.arff -- -P 100 -S 1 -I ${iter} -W
weka.classifiers.trees.DecisionStump  &>> CostSensitive_AdaBoost_DecisionStump.txt


done


done


exit 0
```

## VI. Appendix B – Fit Data FP rates

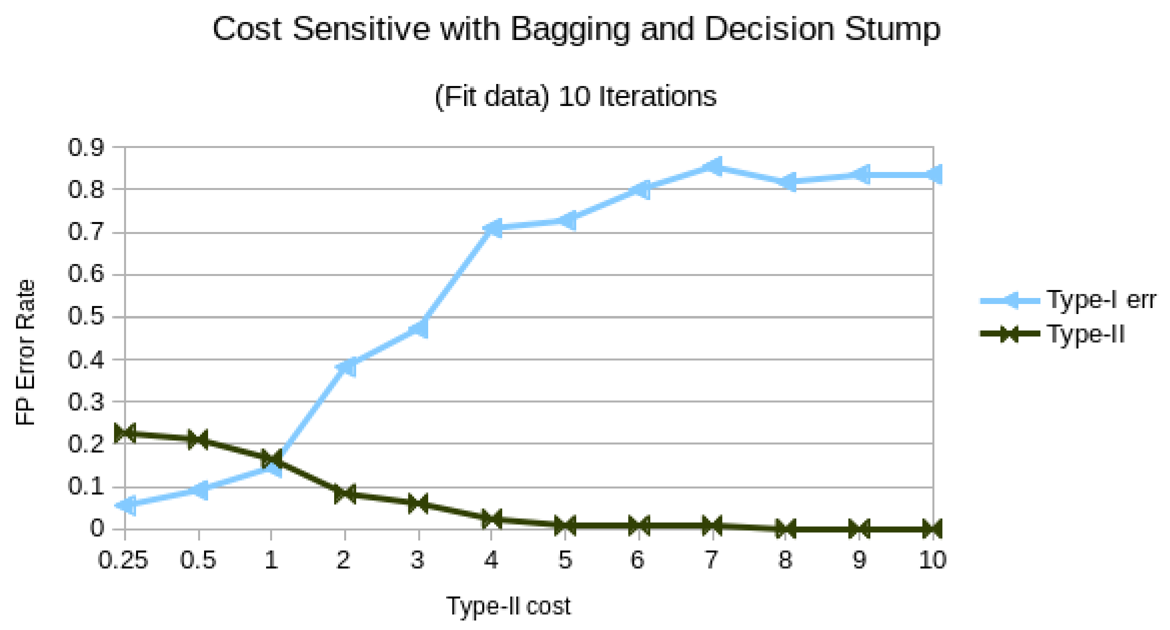
Graphs comparing the Fit data are included here for reference only.

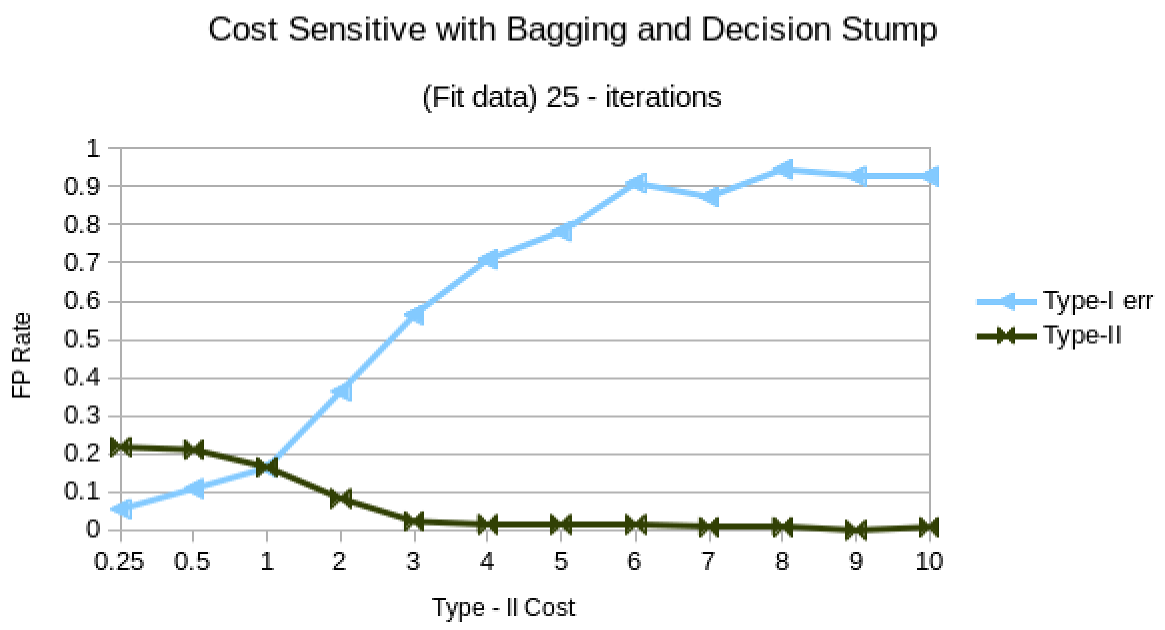
### Cost sensitive classifier combined with bagging and J48



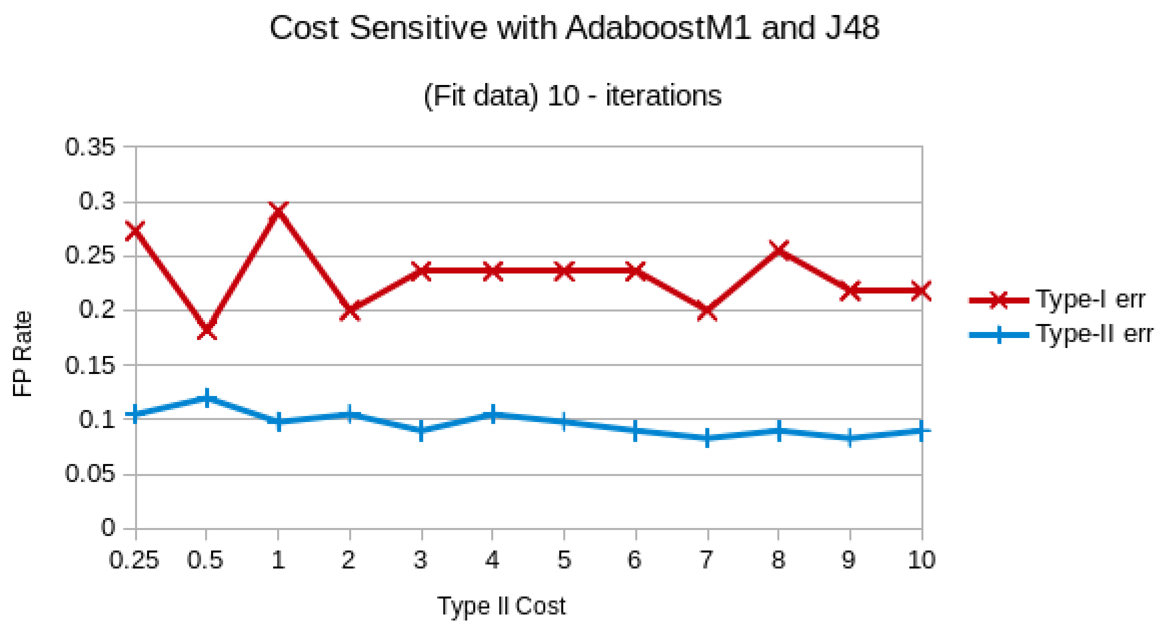


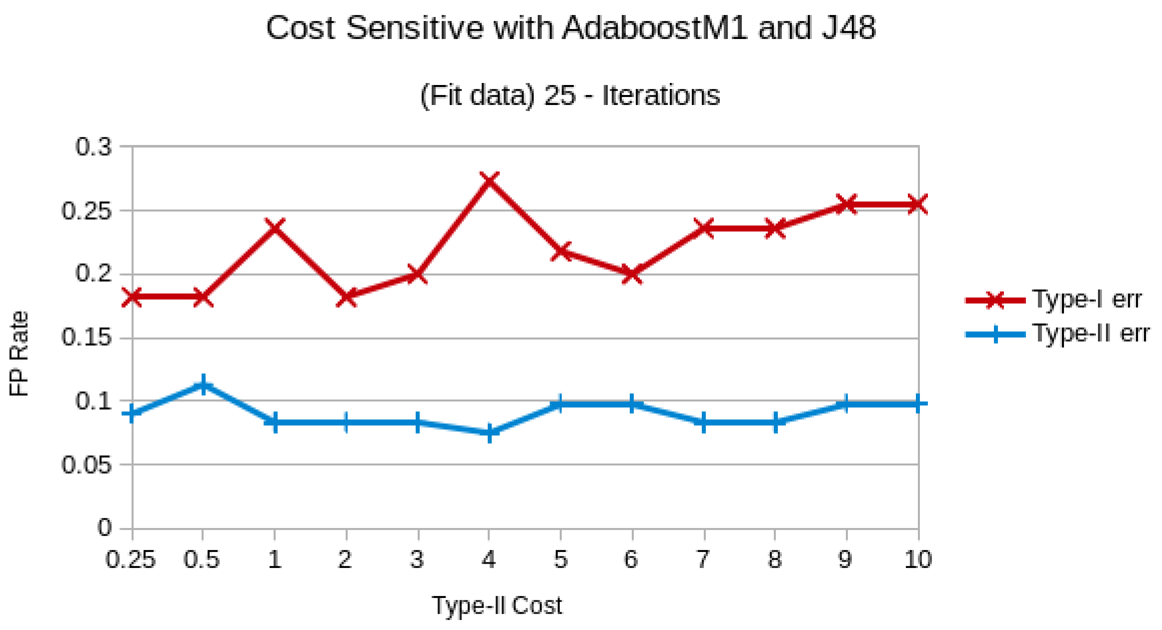
## Cost sensitive classifier combined with bagging and Decision Stump



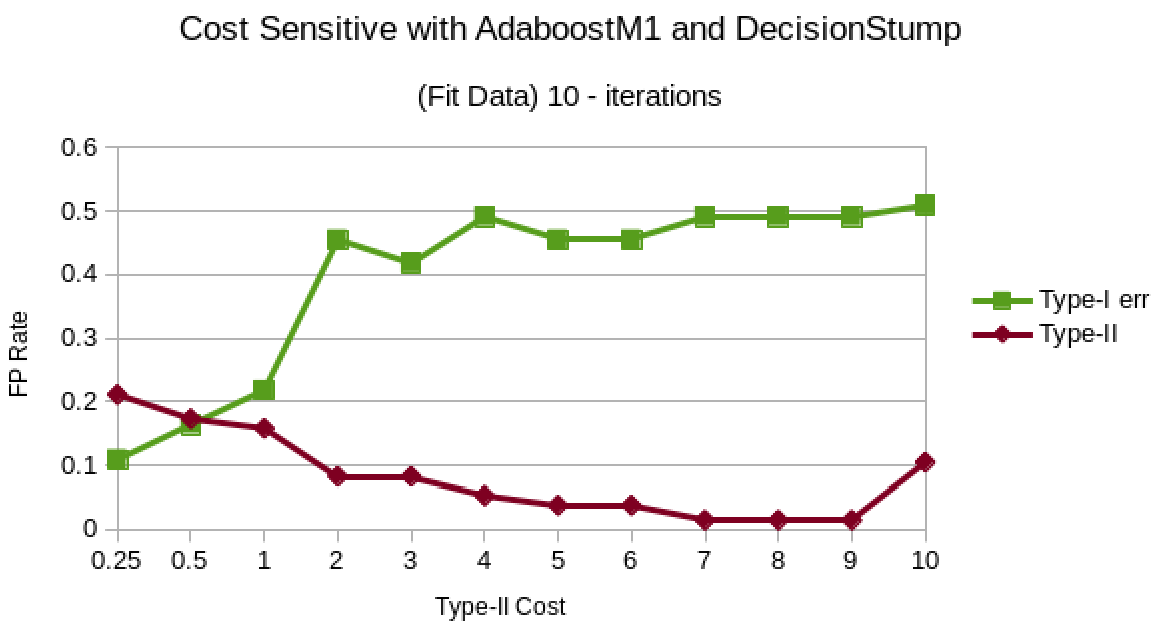


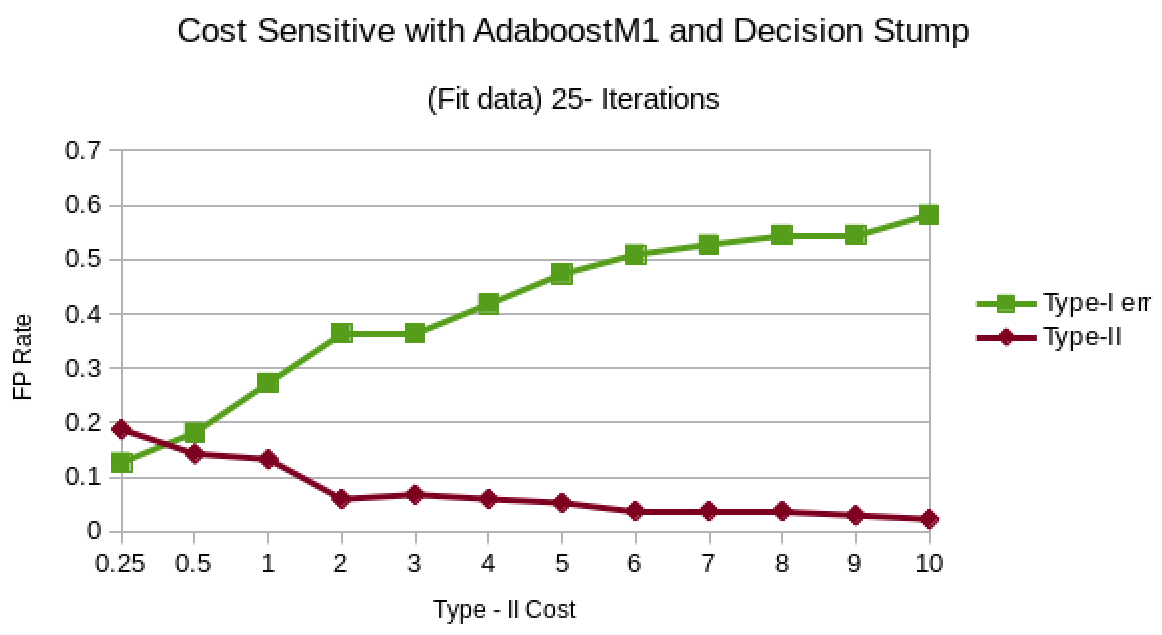
## Cost sensitive classifier combined with boosting (AdaBoostM1) and J48





## Cost sensitive classifier combined with boosting (AdaBoostM1) and Decision Stump



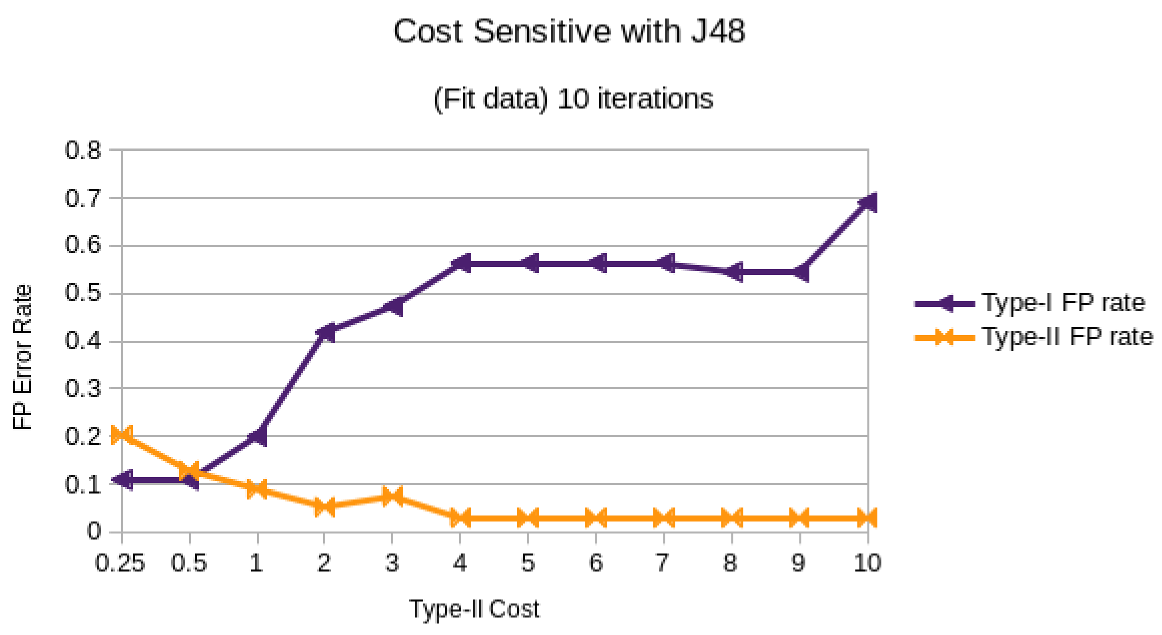


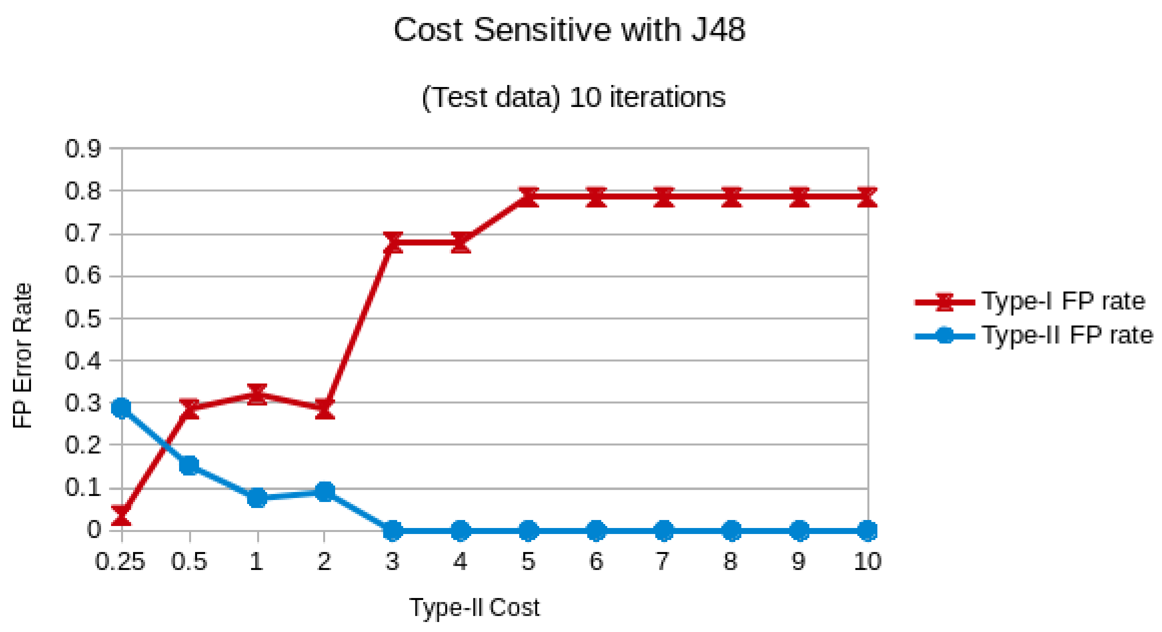
## VII. Appendix C – FP rates from HW2.

The FP rates from a prior assignment are included for reference only. Of particular note is the size of the decision tree reaches a maximum when the Type-I and Type-II FP Rates are nearly equal.

| **Type-II cost** | 10 – Iterations | | | | Decision Tree | | | |
|---|---|---|---|---|---|---|---|---|
| | Fit | | Test | | Fit | | Test | |
| | Type-I | Type-II | Type-I | Type-II | Leaves | Size | Leaves | Size |
| 0.25 | 0.109 | 0.203 | 0.036 | 0.288 | 2 | 3 | 2 | 3 |
| **0.5** | **0.109** | **0.128** | **0.286** | **0.152** | **3** | **15** | **8** | **15** |
| **1.0** | **0.200** | **0.090** | **0.321** | **0.076** | **8** | **15** | **8** | **15** |
| 2.0 | 0.418 | 0.053 | 0.286 | 0.091 | 2 | 3 | 2 | 3 |
| 3.0 | 0.473 | 0.075 | 0.679 | 0.000 | 3 | 5 | 3 | 5 |
| 4.0 | 0.564 | 0.030 | 0.679 | 0.000 | 3 | 5 | 3 | 5 |
| 5.0 | 0.564 | 0.030 | 0.786 | 0.000 | 3 | 5 | 3 | 5 |

*Table 5: Classification with Decision Tree (J48)*


Cost Sensitive with J48
(Fit data) 10 iterations
FP Error Rate
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.25
0.5
1
2
3
4
5
6
7
8
9
10
Type-II Cost
Type-I FP rate
Type-II FP rate


Cost Sensitive with J48
(Test data) 10 iterations
FP Error Rate
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.25
0.5
1
2
3
4
5
6
7
8
9
10
Type-II Cost
Type-I FP rate
Type-II FP rate