Classification of satisfaction level based on survey questions and features selection using decision trees.

Vishrant Gupta
Northern Illinois University, DeKalb

December 01, 2017

Abstract

This project applies methods for classifying the satisfaction level into 'Happy' and 'Unhappy', and reducing the number of questions asked in a survey. This project deals with questionnaire fielded in the LISS panel [1].

1 Introduction

Feedback is a valuable information for timely improvement of a system and for making important decisions. A feedback can be positive, constructive or negative. Investigators might have multiple questions, however it is often cumbersome to answer large set of questionnaires and most of the time large questionnaires is left incomplete, this might results in invalid prediction. This project deals with questionnaire fielded in the LISS panel and aims at classifying satisfaction based on survey questions A and reducing the number of questions that should be asked based on a question significance.

2 Data collection

The dataset for this study was obtained from questionnaires presented to LISS panel. The questionnaires was presented in May 2017, and was repeated in June 2017 for non-respondents. The questionnaires was presented to 7,167 panel members out of which 6,010 members fully completed the questionnaires, response rate of 83.85%. The questionnaire is part of the ninth wave of the LISS Core Study, a longitudinal survey providing a broad range of social core information about the panel members. However data collected have imbalanced classes 3, to create a balanced dataset Synthetic Minority Over-sampling Technique (SMOTE) 4.1 and SpreadSubsample 4.2 is used. The data distribution

[1] `https://www.dataarchive.lissdata.nl/study_units/view/668`

can be seen in figure 1 after class balancing.

As it can be seen in section A that predictors are quantitative and they should be standardized as predictor have different feedback range. Some of the predictors ranges from 1 to 10 and some from 1 to 5. In this project, all the predictors were normalized using Python script to bring them at the same level. For example the response variable cp17i010 ranges from 1 to 10 while predictors cp17i020 - cp17i069 ranges from 1 to 5, thus it was required to standardized the response variable cp17i010 compared with predictors. After standardization, label is assigned to response variable cp17i010, label 'unhappy' is assigned to response variable between 0 to 2.5 and 'happy' is assigned between 2.6 to 5.

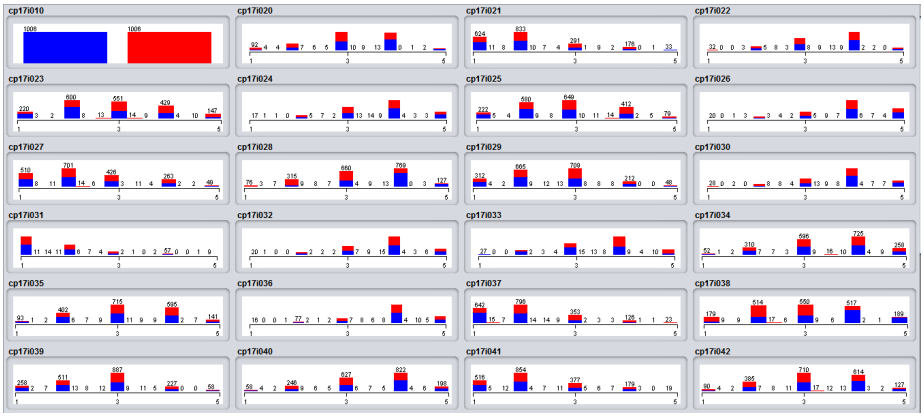

Figure 1: Data distribution

3 Class imbalance problem

Imbalance data refers to the classification problem when the number of sample points in each classes are not equal. For example in figure 2 there are 5030 sample points for class 'Happy' as compare to 843 sample points in 'Unhappy' class. Class imbalance problem creates a bias for a classification algorithm, which may result in a misleading accuracy. Here 85.64% belongs to class 'Happy' and remaining belong to 'Unhappy' so a dummy classifier can give at least an accuracy of 85.64% and thus creates a misleading sense of highly accurate model by under-performing on all other classes.

3.1 Class imbalance problem is common

In most of the real-world classification problem, class imbalance are predominately composed of 'normal' cases as compared with 'abnormal' cases and thus it makes class imbalance a common problem. For example in a dataset for fraudulent transaction the vast majority of transaction will fall under 'Not-fraudulent'

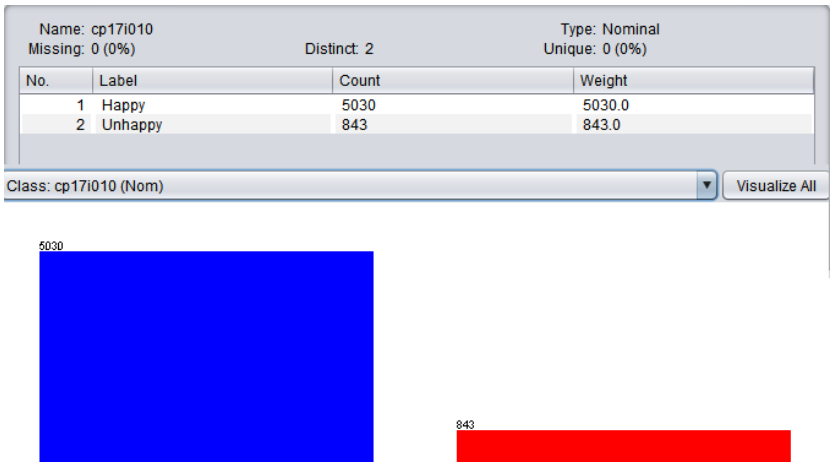

Figure 2: An illustration for class imbalance

while there will be very small sample points for 'Fraudulent' transactions. These abnormal cases can be treated as 'rare' events and it needs to be handled properly. There are multiple techniques that can be opted like:

- If possible *collect more data* for imbalanced class.
- *Resample* dataset.

  There could be two main approach for leveling the classes.

  - You can *over-sample* your dataset, which can be done by copying the instances with replacement
  - You can *under-sample* your dataset, which can be done by randomly deleting the instances until classes are balanced.

- Generate dataset using *Synthetic Samples* [1].

  There are some popular synthetic algorithms which can be used to generate synthetic samples. The most popular is Synthetic Minority Oversampling Technique 4.1 which is used for over-sampling dataset. It works by creating synthetic samples for minor classes instead of creating a copies of it. The algorithms selects near by sample points and generate a new sample point which will be like the collected samples.

- Try Penalized Models

  The penalized models like penalized-SVM and penalized-LDA imposes an additional cost on models for making classification mistakes on the minority class during training. These penalties can bias the model to pay more attention to the minority class.

4 Methodology

4.1 SMOTE: Synthetic Minority Over-sampling Technique [2]

SMOTE is a technique in which minority classes are oversampled by creating a *Synthetic Samples* [1] than simply over sampling existing sample points with replacement. The minority class is over sampled depending upon the k nearest neighbors chosen randomly. Algorithm 3 explains how SMOTE algorithm works.

Consider a sample point (2,3) and it's nearest neighbour (3,4).
(2,3) is a sample ponit
(3,4) is one of it's k nearest neighbours
Let:
f1_1 = 2, f2_1 = 3, f2_1 - f1_1 = 1
f1_2 = 3, f2_2 = 4, f2_2 - f1_2 = 1
Thus, new data point generated will be:
(f1',f2') = (2,3) + rand(0 - 1) * (1, 1)
Where rand(0 - 1) generates a random number between 0 and 1

Table 1: An example for generating a sample point with SMOTE

4.1.1 Combining under-sampling and SMOTE

In order to balance the classes a combination of under-sampling and SMOTE can be used. The majority class is under-sampled by randomly removing sample points from it until it becomes specified percentage of minority class. For example, if there are 25 sample points in minority class and 50 sample points in majority class, then the major class is under-sampled to 25 points. By applying this technique the bias of a model for majority class is now revered to minority class. Nitesh V. Chawla in his paper [2] pointed out that:

> "Combination of over-sampling the minority (abnormal) class and under-sampling the majority (normal) class can achieve better classifier performance (in ROC space) than only under-sampling the majority class. Over-sampling the minority class and under-sampling the majority class can achieve better classifier performance (in ROC space) than varying the loss ratios in Ripper or class priors in Naive Bayes"

4.2 SpreadSubsample

SpreadSubsample produces a random subsample of a dataset. The original dataset must fit entirely in memory. This filter allows you to specify the maximum 'spread' between the rarest and most common class. For example, you

Algorithm *SMOTE*(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE $N\%$; Number of nearest neighbors k
Output: $(N/100) * T$ synthetic minority class samples

1. (∗ *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* ∗)
2. **if** $N < 100$
3. **then** Randomize the T minority class samples
4. $T = (N/100) * T$
5. $N = 100$
6. **endif**
7. $N = (int)(N/100)$ (∗ *The amount of SMOTE is assumed to be in integral multiples of 100.* ∗)
8. k = Number of nearest neighbors
9. $numattrs$ = Number of attributes
10. $Sample[\,][\,]$: array for original minority class samples
11. $newindex$: keeps a count of number of synthetic samples generated, initialized to 0
12. $Synthetic[\,][\,]$: array for synthetic samples
(∗ *Compute k nearest neighbors for each minority class sample only.* ∗)
13. **for** $i \leftarrow 1$ **to** T
14. Compute k nearest neighbors for i, and save the indices in the $nnarray$
15. Populate(N, i, $nnarray$)
16. **endfor**

Populate(N, i, $nnarray$) (∗ *Function to generate the synthetic samples.* ∗)

17. **while** $N \neq 0$
18. Choose a random number between 1 and k, call it nn. This step chooses one of the k nearest neighbors of i.
19. **for** $attr \leftarrow 1$ **to** $numattrs$
20. Compute: $dif = Sample[nnarray[nn]][attr] - Sample[i][$
21. Compute: gap = random number between 0 and 1
22. $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
23. **endfor**
24. $newindex{+}{+}$
25. $N = N - 1$
26. **endwhile**
27. **return** (∗ *End of Populate.* ∗)

End of Pseudo-Code.

Figure 3: SMOTE algorithm

may specify that there be at most a 2:1 difference in class frequencies. When used in batch mode, subsequent batches are NOT re-sampled. [2]

adjustWeights	Wether instance weights will be adjusted to maintain total weight per class.
randomSeed	Sets the random number seed for subsampling.
distributionSpread	The maximum class distribution spread. (0 = no maximum spread, 1 = uniform distribution, 10 = allow at most a 10:1 ratio between the classes).
debug	If set to true, filter may output additional info to the console.
maxCount	The maximum count for any class value (0 = unlimited).

Table 2: OPTIONS

SMOTE and SpreadSubsample both the techniques were used to the imbalanced dataset as shown in figure 2 for two classes. SMOTE is applied first to the minority class followed by SpreadSubsample with an option of distributionSpread = 1. The result of which is as shown as follows:

Figure 4: Balanced dataset

[2] `http://weka.sourceforge.net/doc.dev/weka/filters/supervised/instance/SpreadSubsample.html`

5 Data Analysis

5.1 Classification

Three classifiers were implemented to classify the data: Random Forest [4], J48 decision tree [5] and Support Vector Machine (SVM) [3]. The model was trained with 10-fold cross validation. The accuracy, precision, recall, and F-measure metrics was calculated with 10 repeated runs to account for the randomness introduced by the sampling procedure, average of all iterations is shown in below table:

Method	Random Forest		J48 Decision Tree		SVM		Logistic Regression	
Classes	Happy	Unhappy	Happy	Unhappy	Happy	Unhappy	Happy	Unhappy
Correctly Classified (%)	79.20	79.20	78.01	78.01	78.25	78.25	73.70	73.70
Precision	0.80	0.79	0.80	0.76	0.79	0.78	0.73	0.73
Recall	0.78	0.80	0.74	0.82	0.77	0.79	0.74	0.73
F-Measure	0.79	0.79	0.77	0.79	0.78	0.78	0.73	0.73

Table 3: Accuracy, Precision, Recall and F-Measure for different models used

It can be concluded that Random Forest performed the best with 79.20% of accuracy followed by J48 decision tree with an accuracy of 78.01%. But Random Forest use of all the predictors as compared to J48 decision where it uses part of the predictors and is fast as compared to random forest.

5.2 Feature reduction

One of the key in forming effective feedback is asking significant question to make future decisions. In the section 5.1 we concluded that random forest performed well for classifying sample points but decision tree can be used to get the most important features [6] and the number of features can be controlled by restricting the number of sample points in each feature spaces formed using decision tree. In figure 5 there are 8 feature spaces with different deciding predictors for each feature space, but decision tree after pruning have chosen 7 predictors out of 50 as the significant predictors where root is the most important one. N. M. Tahir in his paper [6] stated that:

> "Using decision tree (DT) as a classifier, two distinct processes need to be implemented namely, building the tree and then performing classification using the top-down approach in DT construction. At each node, decision to what is likely to be the best split is determined using the predicted value. Each branch in the tree is labeled with its decision rule whilst each terminal node is labeled with the predicted value of that node. Cross validation ensures selection of an optimum tree size and helps avoid the problem of over fitting. Consequently, the framework has enabled us to select the best and most optimized eigenpostures for classification purpose."

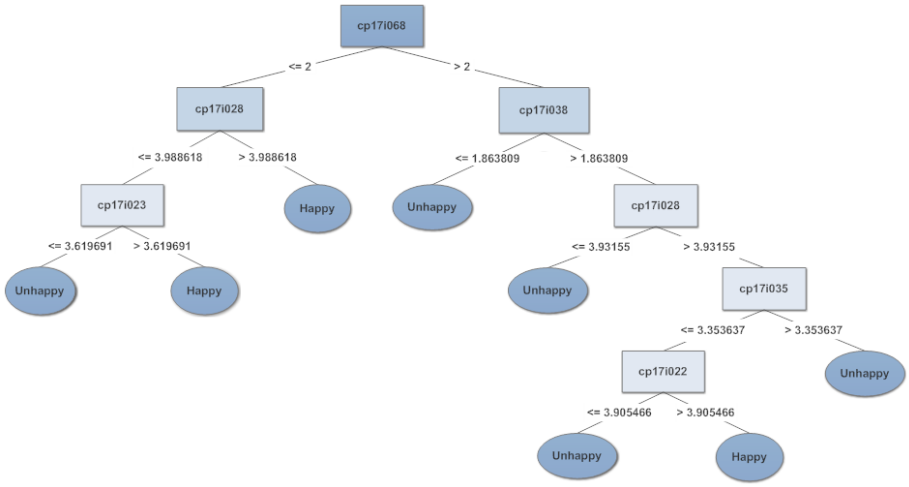

Figure 5: J48 pruned tree

The scatter plot for most significant predictor is shown in figure 6. It can be seen that most of the blue spots which belong to class happy is when value of predictor cp17i068 A is below 2, however model also depends on predictor cp17i023 i.e. when cp17i023 is less than or equal to 3.619691 the selected class is unhappy.

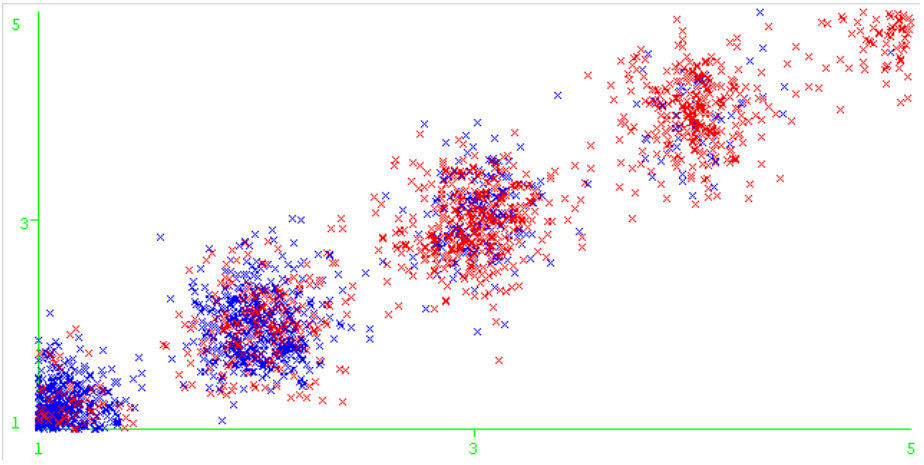

Figure 6: cp17i068 scatter plot

6 Conclusion

This project deals with questionnaires presented to LISS panel and the problem of long questionnaires and asking important question was solved using the decision tree with an accuracy of 78.01% and reducing the number of questions from 50 to 7 questions. The random forest however showed an improvement of 1%, however it uses all the predictors to build a classification model. Thus as compared to random forest J48 decision tree have performed well taking into consideration the number of predictors it have used which is 7, the result of J48 decision tree can be seen in figure 5. Other models like logistic regression, NaiveBayes were also tried, however random forest and decision tree have performed better then those methods. Future work includes providing the output (considering only those predictors selected using decision tree) of decision tree to neural network using tensor flow so the accuracy can be further improved.

Appendix A Data interpretation

The concept of every predictor is as below:

cp17i010: happiness (ESS)
cp17i011: satisfaction (POLS, LSO)
cp17i012 - cp17i013: mood (state, trait)
cp17i014 - cp17i018: satisfaction with life (Diener)
cp17i019: trust (ESS)
cp17i020 - cp17i069: BIG-V (IPIP, Goldberg)

Table 4: Data interpretation

cp17i010
On the whole, how happy would you say you are?
0 0 totally unhappy
1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
10 10 totally happy
999 I dont know

cp17i020 - cp17i029
Please use the rating scale below to describe how accurately each statement describes you.
cp17i020 Am the life of the party.
cp17i021 Feel little concern for others.
cp17i022 Am always prepared.
cp17i023 Get stressed out easily.
cp17i024 Have a rich vocabulary.
cp17i025 Dont talk a lot.
cp17i026 Am interested in people.
cp17i027 Leave my belongings around.
cp17i028 Am relaxed most of the time.
cp17i029 Have difficulty understanding abstract ideas.
1 very inaccurate
2 moderately inaccurate
3 neither inaccurate nor accurate
4 moderately accurate

5 very accurate

cp17i030 - cp17i039
Please use the rating scale below to describe how accurately each statement describes you.
cp17i030 Feel comfortable around people.
cp17i031 Insult people.
cp17i032 Pay attention to details.
cp17i033 Worry about things.
cp17i034 Have a vivid imagination.
cp17i035 Keep in the background.
cp17i036 Sympathize with others feelings.
cp17i037 Make a mess of things.
cp17i038 Seldom feel blue.
cp17i039 Am not interested in abstract ideas.
1 very inaccurate
2 moderately inaccurate
3 neither inaccurate nor accurate
4 moderately accurate
5 very accurate

cp17i040 - cp17i049
Please use the rating scale below to describe how accurately each statement describes you.
cp17i040 Start conversations.
cp17i041 Am not interested in other peoples problems.
cp17i042 Get chores done right away.
cp17i043 Am easily disturbed.
cp17i044 Have excellent ideas.
cp17i045 Have little to say.
cp17i046 Have a soft heart.
cp17i047 Often forget to put things back in their proper place.
cp17i048 Get upset easily.
cp17i049 Do not have a good imagination.
1 very inaccurate
2 moderately inaccurate
3 neither inaccurate nor accurate
4 moderately accurate
5 very accurate

cp17i050 - cp17i059
Please use the rating scale below to describe how accurately each statement describes you.
cp17i050 Talk to a lot of different people at parties.

cp17i051 Am not really interested in others.
cp17i052 Like order.
cp17i053 Change my mood a lot.
cp17i054 Am quick to understand things.
cp17i055 Dont like to draw attention to myself.
cp17i056 Take time out for others.
cp17i057 Shirk my duties.
cp17i058 Have frequent mood swings.
cp17i059 Use difficult words.
1 very inaccurate
2 moderately inaccurate
3 neither inaccurate nor accurate
4 moderately accurate
5 very accurate

cp17i060 - cp17i069
Please use the rating scale below to describe how accurately each statement describes you.
cp17i060 Dont mind being the center of attention.
cp17i061 Feel others emotions.
cp17i062 Follow a schedule.
cp17i063 Get irritated easily.
cp17i064 Spend time reflecting on things.
cp17i065 Am quiet around strangers.
cp17i066 Make people feel at ease.
cp17i067 Am exacting in my work.
cp17i068 Often feel blue.
cp17i069 Am full of ideas.
1 very inaccurate
2 moderately inaccurate
3 neither inaccurate nor accurate
4 moderately accurate
5 very accurate

Appendix B Steps to use WEKA [7]

B.1 Preprocessing or cleaning

- Launch Weka → click on the tab Explorer
- Load a dataset. (Click on 'Open File' & locate the datafile)
- Click on PreProcess tab & then look at your lower R.H.S. bottom window click on drop down arrow and choose 'No Class'
- Click on 'Edit' tab, a new window opens up that will show you the loaded datafile. By looking at your dataset you can also find out if there are missing values in it or not. Also please note the attribute types on the column header. It would either be 'nominal' or 'numeric'.

If your data has missing values then its best to clean it first before you apply any forms of mining algorithm to it.

Data cleaning: To clean the data, you apply 'Filters' to it. Generally the data will be missing with values, so the filter to apply is 'ReplaceMissing-WithUserConstant' (the filter choice may vary according to your need, for more information on it please consult the resources).Click on Choose button below Filters → Unsupervised → attribute → 'ReplaceMissingWithUserConstant'

B.2 Classify

Once the preprocessing or cleaning of data is done, you can classify your dataset by choosing required model. The option of choosing model is available with just few clicks. It is as shown in figure 7.

Once you have selected the desired model you can change the options for model as shown in figure 8. Once you change the options accordingly you can start building the model using 'Start' button and the result of which will be shown in 'Classifier output'.

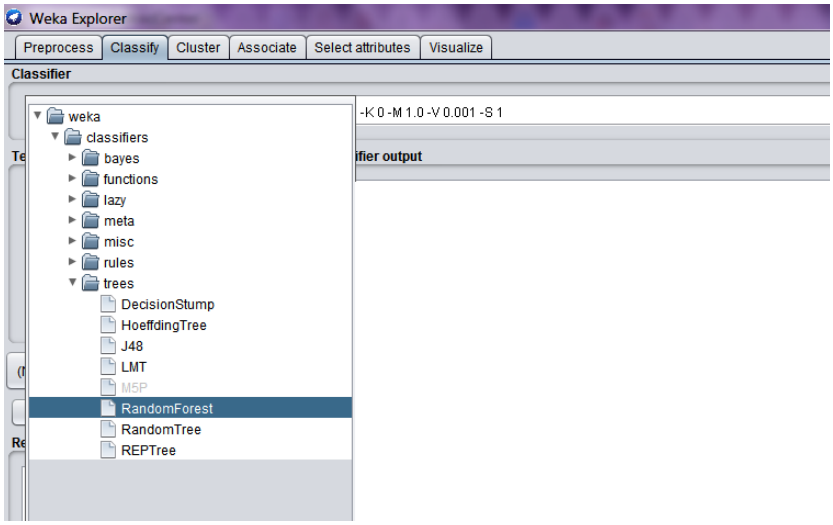

Figure 7: WEKA model selection

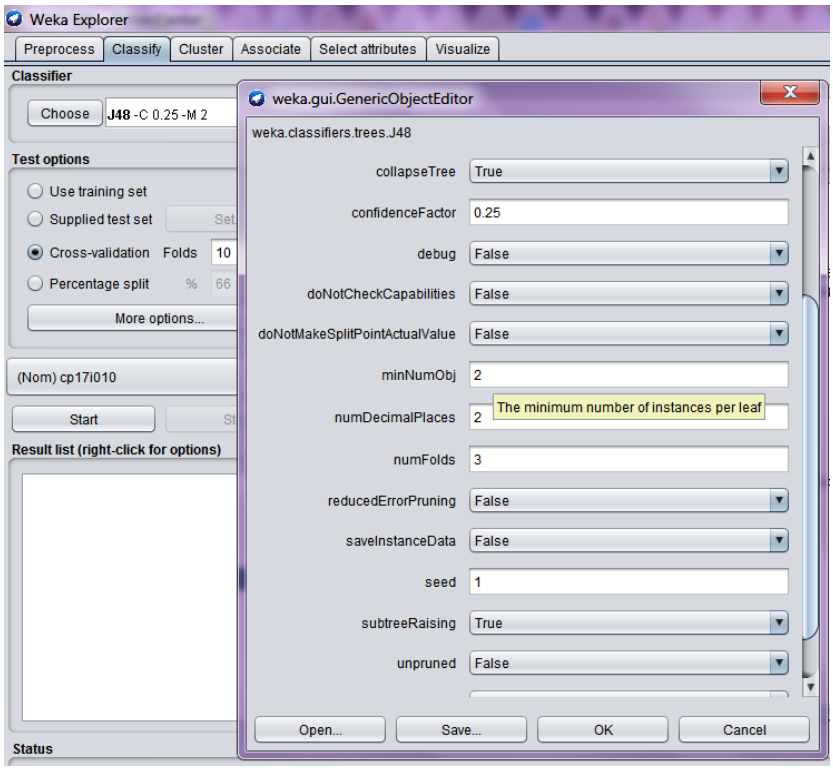

Figure 8: WEKA model option selection

References

[1] E. L. Barse, H. Kvarnstrom, and E. Jonsson. Synthesizing test data for fraud detection systems. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, pages 384–394, Dec 2003.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[3] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[4] T. K. Ho. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, pages 278–, Washington, DC, USA, 1995. IEEE Computer Society.

[5] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[6] N. M. Tahir, A. Hussain, S. A. Samad, K. A. Ishak, and R. A. Halim. Feature selection for classification using decision tree. In *2006 4th Student Conference on Research and Development*, pages 99–102, June 2006.

[7] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 4th edition, 2016.