THE PITFALLS OF OVERFITTING IN OPTIMIZATION OF A MANUFACTURING QUALITY CONTROL PROCEDURE

Tea Tušar
DOLPHIN Team, INRIA Lille – Nord Europe, France
Department of Intelligent Systems, Jožef Stefan Institute, Ljubljana, Slovenia
tea.tusar@ijs.si

Klemen Gantar
Faculty of Computer and Information Science, University of Ljubljana, Slovenia
kg6983@student.uni-lj.si

Bogdan Filipič
Department of Intelligent Systems, Jožef Stefan Institute, Ljubljana, Slovenia
Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
bogdan.filipic@ijs.si

Abstract We are concerned with the estimation of copper-graphite joints quality in commutator manufacturing—a classification problem in which we wish to detect whether the joints are soldered well or have any of the four known defects. This quality control procedure can be automated by means of an on-line classifier that can assess the quality of commutators as they are being manufactured. A classifier suitable for this task can be constructed by combining computer vision, machine learning and evolutionary optimization techniques. While previous work has shown the validity of this approach, this paper demonstrates that the search for an accurate classifier can lead to overfitting despite cross-validation being used for assessing the classifier performance. We inspect several aspects of this phenomenon and propose to use repeated cross-validation in order to amend it.

Keywords: Computer vision, Differential evolution, Machine learning, parameter tuning, Manufacturing, Quality control.

1. Introduction

In automotive industry, only one part per million of supplied products is allowed be defective, which yields strict requirements for the involved manufacturing processes as well as their quality control procedures. We are interested in the manufacturing of graphite commutators (i.e., components of electric motors used, for example, in automotive fuel pumps) produced at an industrial production plant. More specifically, we wish to automatically assess the quality of copper-graphite joints in commutators after the soldering phase of this manufacturing process, which is one of the most critical phases of commutator production.

At present, the soldering quality control at the plant is done manually. Automated on-line quality control would bring several advantages over manual inspection. For example, it can promptly detect irregularities making error resolution faster and consequently saving a considerable amount of resources. Moreover, it does not slow down the production line and is cheaper than manual inspection. Finally, it does not suffer from fatigue and other human factors that can result in errors. This is why we aim for an automated on-line quality control procedure capable of determining whether the joints are soldered well or have any of the four known defects.

Such automation can be implemented on the production line with a classifier previously constructed on a database of commutator segment images with known defects (or absence of defects). Three previous studies [3, 4, 5] have already tackled this problem and in all cases 10-fold cross-validation (CV) was used as a measure of classifier accuracy. This work questions 10-fold CV as the measure of choice for such tasks and proposes actions to deal with the inevitable overfitting issue.

The rest of the paper is structured as follows. Section 2 presents details of the problem in question, summarizes previous work and outlines the design of the quality control procedure used in this study. Section 3 is devoted to cross-validation and the overfitting issue. Performed experiments and their results are discussed in Section 4. Finally, Section 5 summarizes the paper and gives ideas for future work.

2. Background

2.1 Soldering in Commutator Manufacturing

The soldering phase in the commutator manufacturing process consists of soldering the metalized graphite to the commutator copper base. The quality of the resulting copper-graphite joints is crucial since the reliability of end user applications directly depends on the strength of

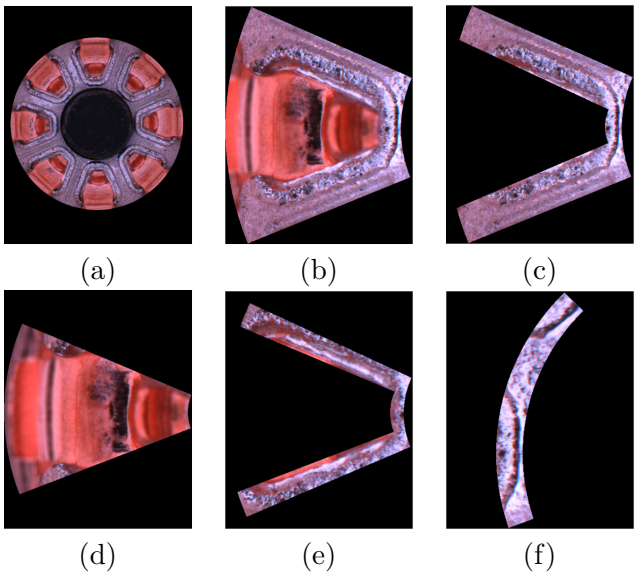

Figure 1: Images of: (a) a graphite commutator, (b) a commutator segment, (c) a ROI for metalization defect, (d) a ROI for excess of solder, (e) a ROI for deficit of solder, and (f) a ROI for disorientation.

these joints. After the soldering phase, the commutators are manually inspected for presence of any defects. Known defects comprise *metalization defect* (presence of visible defects on the metalization layer), *excess of solder* (presence of solder spots on the copper pad), *deficit of solder* (lack of solder in the graphite-copper joint) and *disorientation* (disorientation between the copper body and the graphite disc). Commutators are made up of a number of segments, depending on the model (the considered commutator model from Fig. 1 (a) consists of eight segments). If a single segment has any of the listed defects, the whole commutator is labeled as defective and removed from the production process.

Various defects occur in different regions of the commutator segment. For example, the region where the excess of solder is usually detected is different from the region where disorientation can be observed. Therefore, images of commutator segments can be divided into four regions of interest (ROIs), one for each defect (see Fig. 1).

Because five different outcomes are possible (rare cases where two or more defects appear on a single commutator segment are labeled with just one defect and are not differentiated further), we treat this as a classification problem with five classes. While the manufacturers are indeed interested in keeping statistics of the detected defects, their main concern is that no *false positives* are found. This means that cases when

a defective commutator is labeled as without defects are to be avoided as much as possible. This is, of course, very hard to achieve.

2.2 Previous Work

Three previous studies investigated different aspects of this challenging real-world problem. The initial experiment [4] explored whether computer vision, machine learning and evolutionary optimization techniques could be employed to find small and accurate classifiers for this problem. First, images of the copper-graphite joints were captured by a camera. Next, a fixed set of features were extracted from these images using digital image processing methods. This data were then used to train decision trees capable of predicting if a commutator segment has any of the four defects or is soldered well. The DEMO (Differential Evolution for Multiobjective Optimization) algorithm [12] was applied to search for small and accurate trees by navigating through the space of parameter values of the decision tree learning program. The study found this setup to be beneficial, but urged to focus future research on more sophisticated extraction of features from the images as this seemed to hinder the search for more accurate classifiers.

The second study [5] presented a different setup for the automated quality control procedure to address the issues from the first study. Instead of optimizing decision tree parameter values, differential evolution (DE) [13] was used to search for the best settings of image processing parameters such as filter thresholds. Tuning of these parameters can be a tedious task prone to bias from the engineers that usually do it by trial-and-error experimentation. Moreover, the choice of right features is crucial for obtaining a good classifier.

The single classification problem with five classes was split into four binary classification subproblems, where each subproblem was dedicated to detecting one of the four defects and used data only from the corresponding ROI. In addition, instead of classification accuracy, the measure to be optimized was set to a function penalizing the portion of false negatives 100 times harder than the portion of false positives. The study found that the new combination of computer vision, machine learning and evolutionary optimization techniques was powerful and achieved some good results. While optimization with DE always found better parameter settings for image processing methods than those defined by domain experts, some subproblems proved to be harder than others. For example, detection of commutator segments with excess of solder achieved a satisfactory accuracy, while the detection of metalization defects did not.

The third study [3] investigated the correctness of the implicit assumption from [5] that only features of the subproblem-specific ROI would influence the outcome of the classifier for that subproblem. The study found that features from other ROIs can be important as well, suggesting that it might be better not to split the classification problem into subproblems at all.

While being otherwise rather different, all three mentioned studies used 10-fold CV to estimate the performance of the employed classifiers. In this paper we wish to test if such evaluation of classifiers is appropriate when performing optimization based on this measure.

2.3 Design of the Automated Quality Control Procedure

The automated quality control procedure considered in this paper is very similar to the one presented in [5]. Again computer vision, machine learning and evolutionary optimization methods are combined in the search for the best settings for image processing parameters. In short, the procedure design consists of the following steps:

1 Determine a set of image features.

2 Use an evolutionary algorithm to search for the values of image processing parameters that result in the highest fitness. Evaluate each solution using these steps:

   (a) Based on the chosen parameter values, use the image processing methods to convert each image of a commutator segment into a vector of feature values.

   (b) Construct a classifier (in our case a decision tree) where the vectors of feature values serve as learning instances. Estimate the classifier accuracy and use this value as solution fitness.

3 Choose the best found classifier and the corresponding image processing parameter values to detect defects in images of new commutator segments as they are being manufactured.

Let us now describe the steps of processing commutator segment images, building decision trees and optimizing classifier performance in more detail.

2.3.1 Processing commutator segment images. Processing of images is the most time-consuming task of our procedure and is done in several steps. First, the image of a commutator segment needs to

be properly aligned. Next, the four ROIs shown in Fig. 1 need to be detected. This is done by applying four predefined binary masks to the image, one for each ROI. Each of the ROIs is further processed as follows. Depending on the ROI, the image in RGB format is converted into a gray-scale image by extracting a single color plane. Based on expert knowledge, red is used for all ROIs except the ROI for excess of solder, which uses the blue color plane.

The final three steps require certain parameters to be set. A 2D median filter of size 1×1, 3×3 or 5×5 is applied to reduce noise. Next, a binary threshold that can take values from $\{1, 2, \ldots, 256\}$ is used to eliminate irrelevant pixels. Finally, an additional particle filter is employed to remove all particles (connected pixels with similar properties) with a smaller number of pixels than a threshold value from $\{1, 2, \ldots, 1000\}$. Note that because of the diversity of the defects, it is reasonable to assume that these three image processing parameters should be set independently for each ROI. This means that in total 12 image processing parameters need to be set.

After these image processing steps, the chosen set of features are extracted from the image of each ROI. We use the same set of features as in [5, 3]:

- number of particles,
- cumulative size of particles in pixels,
- maximal size of particles in pixels,
- minimal size of particles in pixels,
- gross/net ratio of the largest particle,
- gross/net ratio of the cumulative size of particles.

To summarize, computer vision methods are used to convert each commutator segment image into a vector of 24 feature values.

2.3.2 Building decision trees. Commutator segment images with known classes are used to construct a database of instances, upon which a machine learning classifier can be built. We chose decision trees since they are easy to understand and implement in the on-line quality control procedure. In accordance with the guidelines from [3], we do not split the machine learning problems into subproblems, but use all instances and all ROIs to build a single classifier with five classes: no defect, metalization defect, excess of solder, deficit of solder and disorientation.

Note that the classifier predicts defects on commutator segments. For the final application, predictions for all segments of a commutator need

to be aggregated in order to produce a prediction for the commutator as a whole. While this might be straightforward to do, it is not the focus of this paper. We first wish to find good classifiers on the segment level before dealing with any meta-classifiers.

2.3.3 Optimizing classifier performance. Classifier performance can be measured in several ways, ranging from classification accuracy to the F-measure to other, even custom functions that depend on the domain (as was done for the two-class case in [5]). While we acknowledge that a similar custom function would be beneficial also for our five-class problem, where false 'no defect' classifications bear more serious consequences than other types of misclassifications, classification accuracy is chosen for now, since it is easier to interpret. Classification accuracy is estimated with 10-fold CV, which is a popular technique for predicting classifier performance on unseen instances and has been used also in the three previous studies [4, 5, 3].

In order to find the values of image processing parameters that will result in a classifier with high accuracy, an evolutionary algorithm is employed to search in the 12-dimensional space of image processing parameter values.

3. The Pitfalls of Overfitting

When building a classifier, some of the data is used for *training* the classifier, while the rest is used for *testing* its performance. Ideally, we would like both sets to be fairly large, since a lot of data is needed to train a classifier well, and a lot of data is needed to truthfully predict how it will perform on unseen instances. However, in reality, the data is often scarce and certain compromises need to be made.

One of the most popular approaches to estimate classifier performance is *k-fold cross-validation*, where the data is split into k sets of approximately equal cardinality. Next, $k-1$ of the sets are used for training the classifier, while the remaining set is used for testing its performance. This is repeated k times so that each set is utilized for testing exactly once. The average of all performance results is then used to estimate the accuracy of the classifier built on the entire data.

This and other cross-validation techniques (see [1] for a survey) were envisioned in order to avoid overfitting, i.e., constructing classifiers that describe noise in the data instead of the underlying relationships, since a classifier that overfits the training data performs poorly on unseen instances. This happens, for example, if the classifier is too complex. However, it has been long known [6] that there exists another source of overfitting that takes place despite cross-validation—if we compare a

high number of classifiers on a small set of instances, the best ones are usually those that overfit these instances.

This means, for example, that if an optimization algorithm that can produce thousands or even millions of solutions is used to find the best classifier for a problem, this best found classifier almost surely overfits the test data. Note however, that our study does not compare multiple classifiers on the same data. Our optimization problem resembles more that of feature selection where a subset of features needs to be found so that classifiers using these features will achieve good performance. The main difference to the standard feature selection is that we are performing feature selection in subgroups—for each of the 12 image processing parameters (one subgroup) we need to select exactly one among all possible values. The danger of overfitting despite using cross-validation has been noticed for feature selection problems as well [11].

In the following we present the results achieved with 10-fold CV for estimating classifier performance on our optimization problem, which show overfitting patterns, and analyze increased pruning and repeated cross-validation as possible alternatives to amend this issue.

4. Experimental Study

4.1 Experimental Setup

The experiments were performed on the commutator soldering domain from previous studies that contains 363 instances with uneven distribution of classes (see Table 1).

Table 1: The commutator soldering domain.

Class	Number of instances	Frequency [%]
No defect	212	58.4
Metalization defect	35	9.6
Excess of solder	35	9.6
Deficit of solder	49	13.5
Disorientation	32	8.8
Total	363	100.0

All computer vision methods were implemented using the Open Computing Language (OpenCL) [7], or more precisely, the OCL programming package [8], an implementation of OpenCL functions in the Open Computer Vision (OpenCV) library [9].

The decision trees were built using the J48 algorithm from the Weka machine learning environment [16], which is a Java implementation of Quinlan's C4.5 decision tree building algorithm [10]. The trees were constructed with default J48 parameter values except for the increased pruning case (more details are given in Section 4.3).

For optimization we use a self-adaptive DE algorithm called jDE [2] with a population of 80 solutions. The stopping criterion for the algorithm was set to 1000 generations. For each set of experiments nine runs have been performed and all presented results show the average values over the nine runs.

4.2 Results of Single Cross-Validation

First, we look at what happens when single 10-fold CV is used to estimate classifier accuracy (see top plot in Fig. 2). The black line shows that jDE is able to find increasingly more accurate classifiers as the evolution progresses. In order to check if these classifiers present signs of overfitting, we perform the following additional assessment. For each run and each best classifier from the population, we estimate the classifier using 10-fold CV ten more times. The span of these accuracies averaged over the nine runs is presented in red.

The increasing gap between the black line and the red area means that classifiers that are good on the default split of instances into cross-validation sets perform considerably worse when they are tested again on ten different cross-validation splits, i.e., the classifiers overfit the default cross-validation split. This happens because we are exploring a large number of classifiers and incidentally optimize them also with regard to the default cross-validation split.

Note that this kind of overfitting is different to the 'usual' one, where the classifier overfits the given instances. While we are probably experiencing both, we cannot know about the second one without testing the classifiers on a large number of unseen instances, which we unfortunately do not posses. We have experimented with reserving a small part of data for validation purposes as was done in [14], but found that this approach is not suitable for our case because of the small number of instances at our disposal. Since we have five classes with uneven distribution of instances, it proved very difficult to find representative instances for validation. Without a representative validation set the resulting estimation of overfitting can be too biased to rely on.

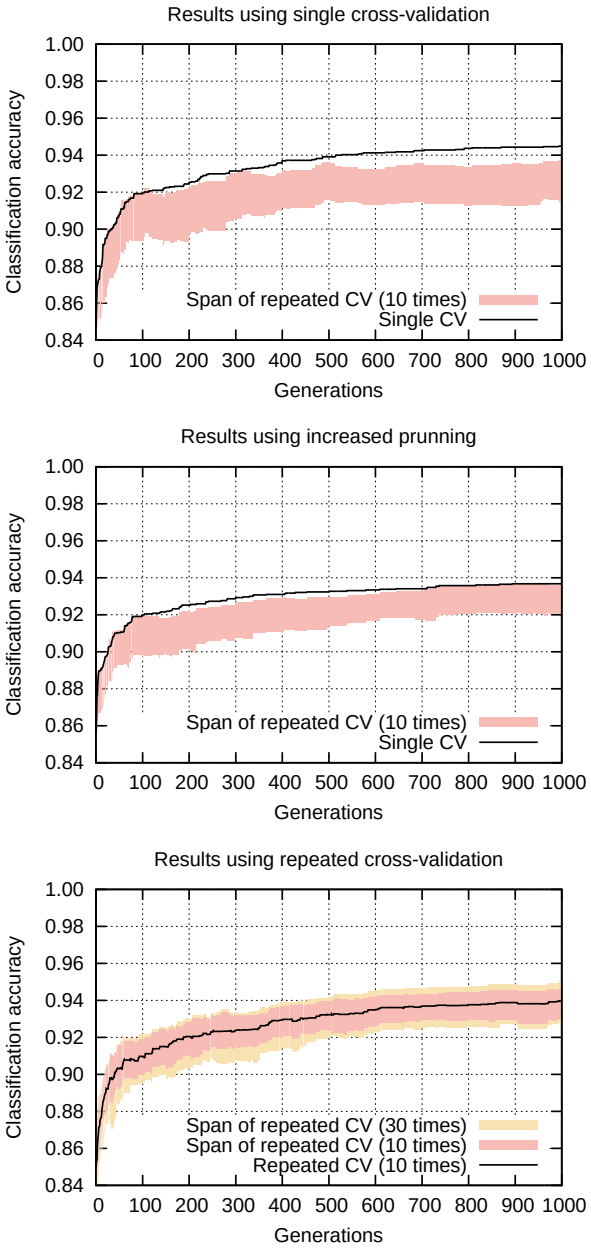

Figure 2: Experimental results of single cross-validation (top plot), increased pruning (middle plot) and repeated cross-validation (bottom plot). The black line shows the best classification accuracy found by jDE, while the red and yellow ares denote the span of accuracy values when the current-best classifier is re-estimated using additional cross-validation. All plots show average values over nine runs.

4.3 Results of Increased Pruning

Next, we investigate whether increased pruning of decision trees can help improve their generalization ability in our case. Note that the original decision trees (the J48 trees with default parameter settings) used in the previous experiments were also pruned. Here, we intensify the pruning by increasing the m parameter of the J48 algorithm that defines the minimal number of instances in any tree leaf from 2 to 5. The results of these experiments are presented in the middle plot in Fig. 2.

While the gap between the single cross-validation and the re-estimation using repeated cross-validation is smaller than in the previous experiments, the overfitting is still obvious. We can conclude that increased pruning does not alleviate much the overfitting brought by optimization.

4.4 Results of Repeated Cross-Validation

Finally, we explore the case when the fitness of the decision trees built with default parameter values is determined as the average of 10 different assessments by 10-fold CV. Again, we perform an additional assessment of the classifiers. This time we add to the repeated cross-validation 20 new estimations (for a total of 30) to see how they compare. The bottom plot in Fig. 2 shows there are no big differences when additional cross-validation results are added, indicating that repeated cross-validation is less prone to overfitting brought by optimization than single cross-validation. The average accuracy of the current-best classifiers over 10 and 30 repetitions are very similar, which suggests 10 repetitions can be chosen over 30 as they require less time.

These results seem to contradict the ones presented in [15], however this is not the case. In a series of experiments, [15] compares the estimates of classification accuracy from single 10-fold CV, 10-fold CV repeated 10 times and 10-fold CV repeated 30 times to a simulated 'true' performance of the classifier on unseen data. The results show that although the confidence interval narrows when increasing the number of cross-validation repetitions, this does not necessarily mean that the accuracy estimate will converge to the 'true' accuracy. The authors argue that the reason for this behavior is that the same data is continuously being resampled in repeated cross-validations. The experiments in [15] tackle the 'usual' overfitting problem, which is not the subject of this paper. We are concerned with the overfitting brought by optimization and find that repeated cross-validation can alleviate it.

5. Conclusion

We have presented a challenging real-world problem of estimating the quality of the commutator soldering process. We wish to find a classifier able to distinguish among joints soldered well and those that have one of the four possible defects. The problem is tackled using a combination of computer vision, machine learning and evolutionary optimization methods. In essence, we are searching for parameter settings of computer vision methods that can yield a highly accurate classifier.

Since an optimization algorithm that explores a large number of solutions to this problem is being used, we have been confronted with the problem of overfitting. We performed some experiments that have shown how overfitting can be detected and discussed on possible ways to amend it. From the results we conclude that repeated cross-validation can be used to diminish the overfitting bias brought by optimization. However, this results cannot be generalized to other machine learning problems without additional experiments that include a number of other datasets. This is a task left for future work.

The presented real-world problem is not yet solved and we can see many directions for future work. First, since the accuracies achieved are still not good enough for automotive industry standards, our main focus will be to try to improve on that (possibly by not producing even more overfitting). This can be tried, for example, by choosing other image features in addition to the six we have right now, or by trying more sophisticated classifiers than decision trees. Also, we intend to consider other measures of classifier performance beside accuracy. For example, we could use a specialized aggregation function or try to use a multiobjective approach. Finally, we will have to eventually combine the classifications of individual commutator segments into a single classification of the commutator as a whole.

Acknowledgment: This work was partially funded by the ARTEMIS Joint Undertaking and the Slovenian Ministry of Economic Development and Technology as part of the COPCAMS project (`http://copcams.eu`) under Grant Agreement no. 332913, and by the Slovenian Research Agency under research program P2-0209.

The authors wish to thank Valentin Koblar for valuable support regarding the application domain and computer vision issues, and Bernard Ženko for helpful discussions on machine learning algorithms.

References

[1] S. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79, 2010.

[2] J. Brest, S. Greiner, B. Bosković, M. Mernik, and V. Žumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006.

[3] E. Dovgan, K. Gantar, V. Koblar, and B. Filipič. Detection of irregularities on automotive semiproducts. *Proceedings of the 17th International Multiconference Information Society (IS)*, pages 22–25, 2014.

[4] V. Koblar and B. Filipič. Designing a quality-control procedure for commutator manufacturing. *Proceedings of the 16th International Multiconference Information Society (IS)*, pages 55–58, 2013.

[5] V. Koblar, E. Dovgan, and B. Filipič. Tuning of a machine-vision-based quality control procedure for product components in automotive industry. Submitted for publication, 2014.

[6] A. Y. Ng. Preventing "overfitting" of cross-validation data. *Proceedings of the Fourteenth International Conference on Machine Learning (ICML)*, pages 245–253, 1997.

[7] OpenCL: The open standard for parallel programming of heterogeneous systems. `http://www.khronos.org/opencl/`. Retrieved January 25, 2016.

[8] OpenCL module within the OpenCV library. `http://docs.opencv.org/modules/ocl/doc/introduction.html`. Retrieved January 25, 2016.

[9] OpenCV: Open source computer vision. `http://opencv.org/`. Retrieved January 25, 2016.

[10] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[11] R. B. Rao, G. Fung, and R. Rosales. On the dangers of cross-validation. An experimental evaluation. *Proceedings of the SIAM Conference on Data Mining (SDM)*, pages 588–596, 2008.

[12] T. Robič and B. Filipič. DEMO: Differential evolution for multiobjective optimization. *Lecture Notes in Computer Science*, 3410:520–533, 2005.

[13] R. Storn and K. V. Price. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[14] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Automated selection and hyper-parameter optimization of classification algorithms. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 847–855, 2013.

[15] G. Vanwinckelen and H. Blockeel. On estimating model accuracy with repeated cross-validation. *Proceedings of the 21st Belgian-Dutch Conference on Machine Learning (BeneLearn)*, pages 39–44, 2013.

[16] Weka Machine Learning Project. `http://www.cs.waikato.ac.nz/ml/weka/index.html`. Retrieved January 25, 2016.