MSc. Decision Tree Report

Lin Li, Jianing Qiu, Xinyi Du, Yiming Jing
Course: CO395, Imperial College London

29th July, 2018

1 Background

The classic facial expression recognition systems work on recognizing a discrete set of facial expressions, usually including six 'basic' emotions: anger, fear, happiness, sadness and surprise. Similarly, the recognition system implemented within this CBC utilizes the Decision Tree approach to detect above six emotions using AUs(Action Units) as an intermediate layer of abstraction, instead of directly classifying based on the raw data, i.e. images. As a consequence, there are six decision trees obtained from the same training dataset during each time of training. And the final classification decision, which emotion is represented by a particular set of AUs, will be made using some strategies to combine the predictions of six trained decision trees.

2 Design and Implementation

Generally, the whole machine learning procedure can be divided into five stages: Data Reading, Data Preprocessing, Model Creating, Model Learning and Model Evaluation. Following this division, the program is split into modules below: **DataProcessor**(Reading and Processing), **DecisionTree**(Model Creating), **Trainer**(Model Learning) and **Evaluator**(Model Evaluation) with the **Facs** file as the main program defining the entire running flow required by the manual. Besides, two extra modules, **config** and **TreeVisualization**, are added in order to define the global configurations of the whole program and implement the functionality of visualizing the structure of decision tree respectively.

2.1 Data Processor

Currently, the datasets, both clean and noisy, are supplied as the format of *MAT* file(extension *'.mat'*), which will be loaded utilizing the functionality of *SciPy*[1]. Meanwhile, the y data directly read from the data file will be also encoded to the required form of *one-hot*[2], specifically with each single value of y converted to a 1*6 binary list.

Moreover, a pair of functions, *saveObjects()* and *retrieveObjects()*, are designed to achieve the processes, like saving objects into the persistent storage usually refers to the disk from the memory, brings the convenience to further testing by the markers. To be more specific, during program running, a list of instances of *DecisionTree.class* will be serialized into a binary file[3] named 'decision_trees.pkl' through the module *Pickle*[4]

2.1.1 Cross Validation

The cross-validation mentioned in this module focuses on splitting datasets according to the passed parameters *fold*(selected fold index) and *fold_num*(total amount of folds). For example, given the

[1]SciPy: https://www.scipy.org.
[2]one-hot encoding: https://en.wikipedia.org/wiki/One-hot.
[3]Why binary file: Compared to text file, binary file is normally adopted as serialization file; it has better space utilization, hence the reading and writing efficiency will be better.
[4]Pickle: https://docs.python.org/3/library/pickle.html.

parameters 1(fold) and 10(fold_num), the whole dataset will be divided into ten units and the first one will be returned as the test dataset with left nine units as the training dataset.

2.2 Decision Tree

One single decision tree is assembled by many subtrees through recursion. The first subtree chains the rest to generate the whole decision tree. A DecisionTree class will be implemented, which defines three fundamental properties of each subtree: root attribute, which represents the best attribute calculated by ID3 algorithm; emotion, which refers to the facial emotion the whole decision tree represents; branchs, which is used to chain the following subtrees and is implemented by a python dictionary with two key-value pairs. Two keys refer to yes and no respectively, and their corresponding values refer to either a leaf node or another subtree. Each following subtree is initialized by the same emotion as the parent tree and the best attributes chosen by the ID3 algorithm.

2.2.1 Prediction

Essentially, there are two predictions integrated in this module. One is for a single decision tree to predict the classifications, while the other one is to produce the unified output of the decision trees of six emotions. Especially, different from the naive prediction, our prediction will provide the information related to the prediction path, i.e. from the root to the leaf, which will be useful for further integrating the results of different emotion trees.

2.2.2 Combining

As discussed below(4.2), there are several proposals to combine the results of different emotion trees into one output and the selected strategy is to utilize the accumulated *IG(Information Gain)*. In more details, for the condition with more than one tree activated, i.e. predicting yes, the one with highest accumulated IG will be selected, whereas it will be the one with minimum accumulated IG if none activated.

2.3 Training

Basically, the Decision Tree learning algorithm deploys the top-down greedy search through the space of all possible attribute selections. Specifically, during the learning, the *ID3 algorithm* is performed, calculating the *IG(Information Gain)* value for each single attribute alone and picking up the one with largest value, in order to select the best attribute, i.e. AU, as a new descendant node down the branch of current root.

In practice, the learning algorithm is implemented as recursion beginning from the selection of the root nodes for the whole decision tree and ending at all end branches with leafs, i.e. the certain boolean value indicating the classification result. Concerning each iteration, there are mainly two procedures executed to determine if creating new leaf or new node respectively:

1. a new leaf with the majority value of the labels belonging to the current node will be attached to the current branch and terminate the current iteration if one of following conditions occur: all samples with the same labels, no more attributes available to select or no sub-dataset available for the current branch.

2. if the procedure continues without being terminated by above conditions, the best attribute, with the maximum IG value in the current attributes, will be selected as descendant node down the current branch of the parent node. In addition, the remaining attributes, excluding the selected one, and the corresponding split dataset will be passed to next iteration with the updated decision tree object as its 'current' attributes and dataset.

2.4 Evaluation

After obtaining all the prediction results through combination outcomes, the next step is evaluation concerning a set of techniques, *Confusion Matrix*, *Precision Rate*, *Recall Rate*, *F1 Measure* and *Classification Rate*. Due to the application of 10-fold cross-validation, the input data will be ten copies of predictions and labels with the identical data structure but generated on the different folds. Then they will be assigned into one single confusion matrix, which is the base of further measurements.

2.5 Visualization

A separated TreeVisualization class is implemented to visualize the whole decision tree, in which *matplotlib*[5] is imported. The X-axis limit is set to the width of the tree, which equals to the total number of the leaf nodes, and the Y-axis limit is set to the depth of the tree. The width of subtree (i.e. the number of leaf nodes) is used to adjust the size and position of the parent tree so that overlap can be avoided and the visualization can be adaptive to different trees with different size. Each attribute (i.e. AU, also the root node of each subtree) and leaf node is drawn as a square and a circle respectively in each figure.

3 Performance

3.1 Diagrams of the Six Trees

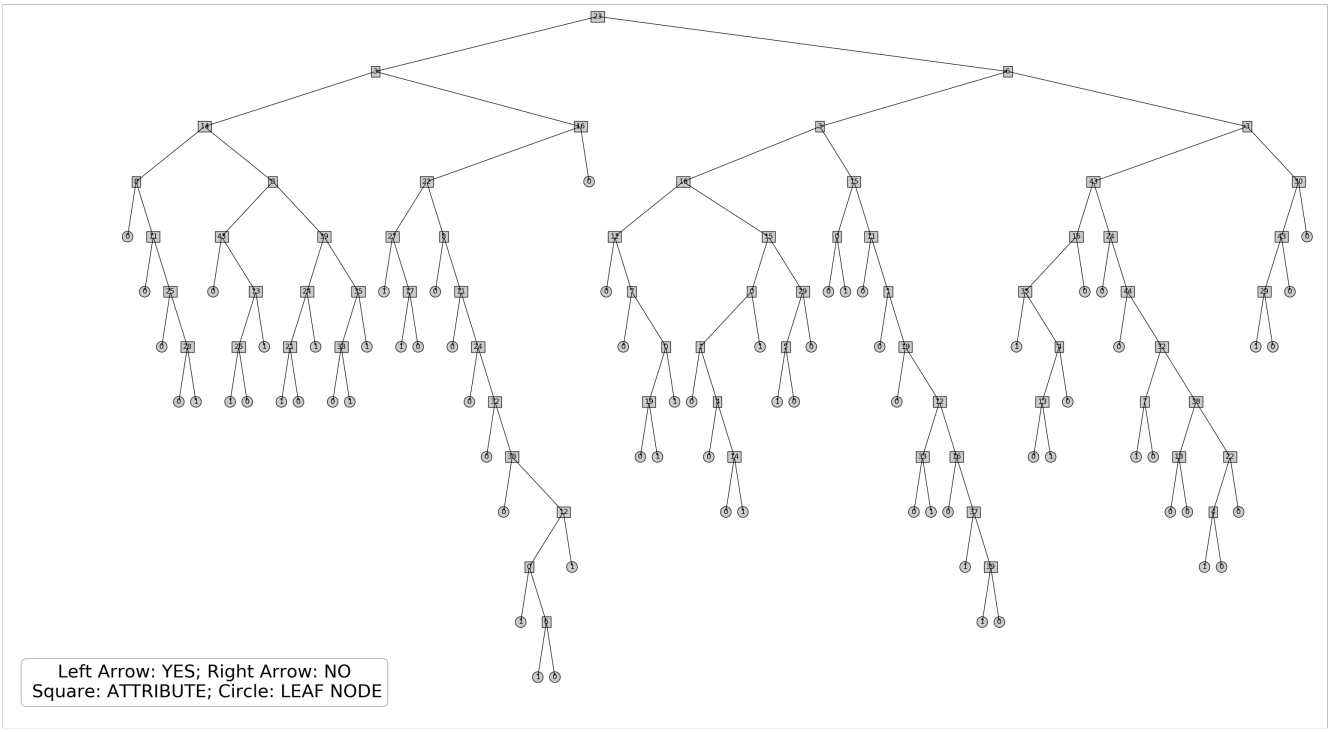

Figure 1: Decision Tree of Anger

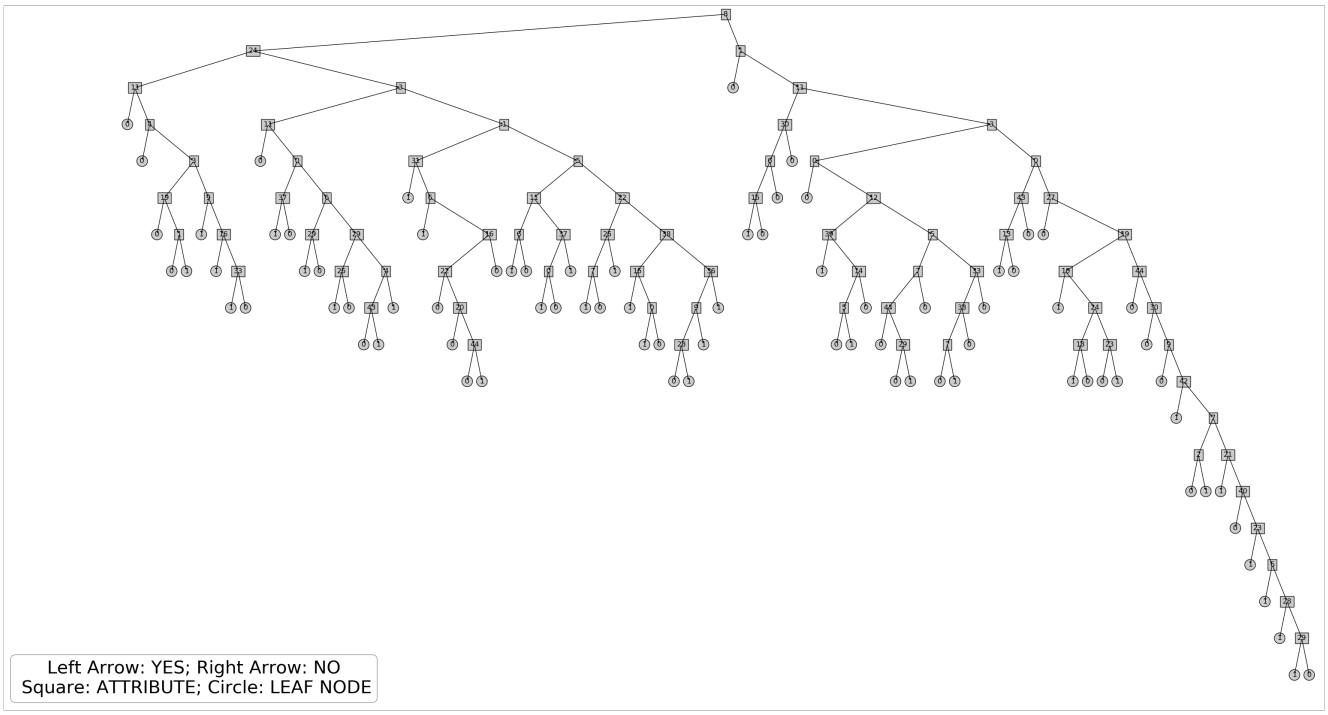

Figure 2: Decision Tree of Disgust

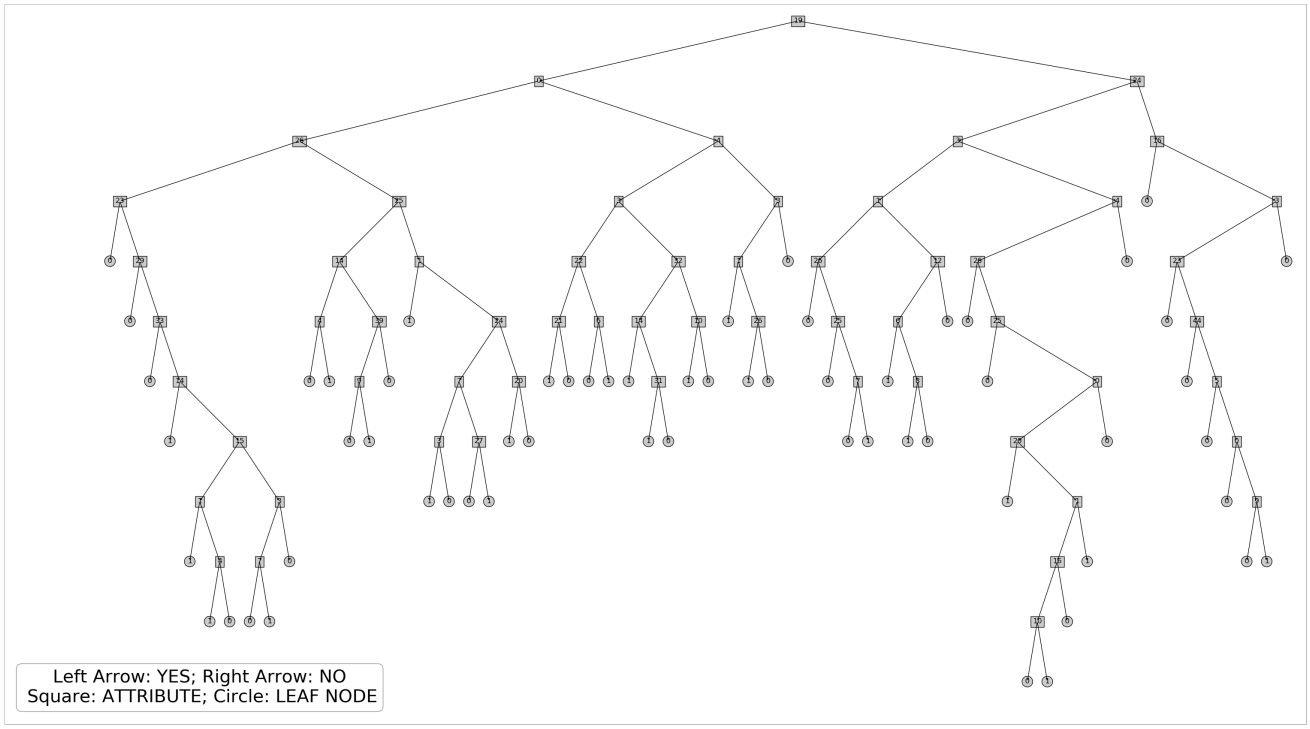

Figure 3: Decision Tree of Fear

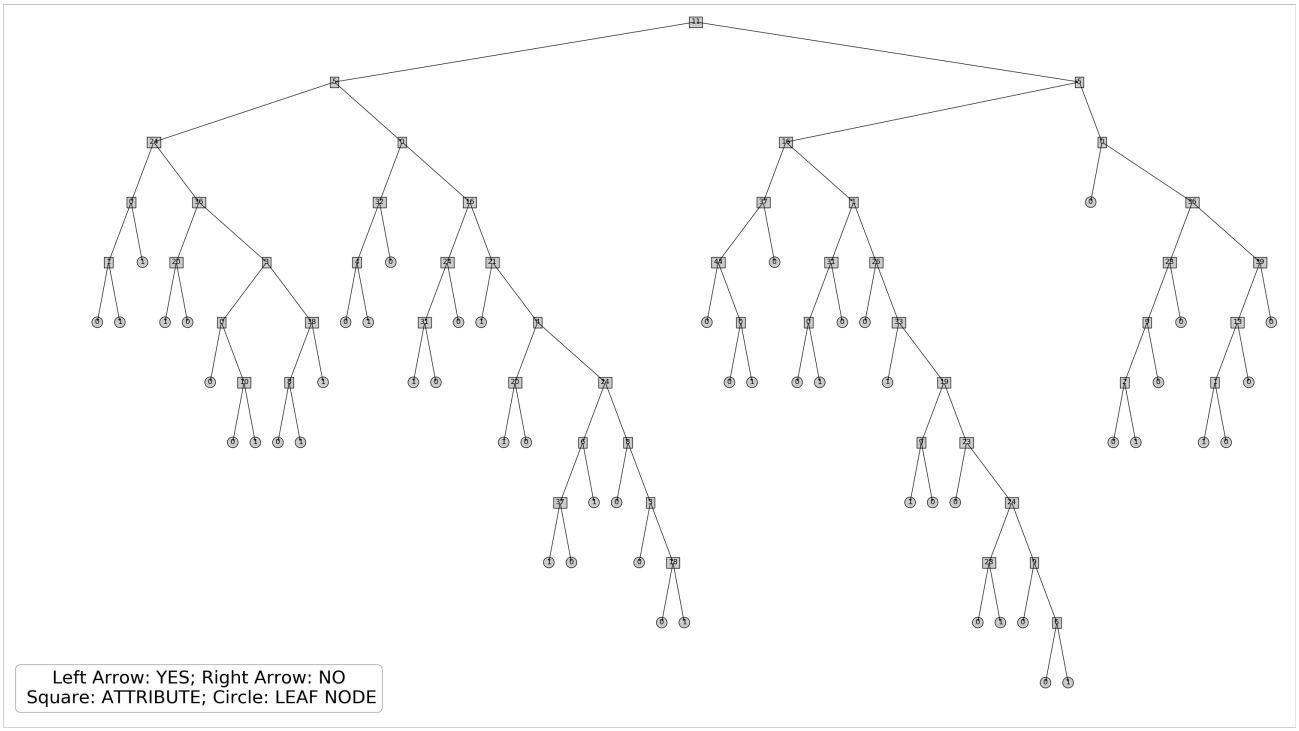

Figure 4: Decision Tree of Happiness

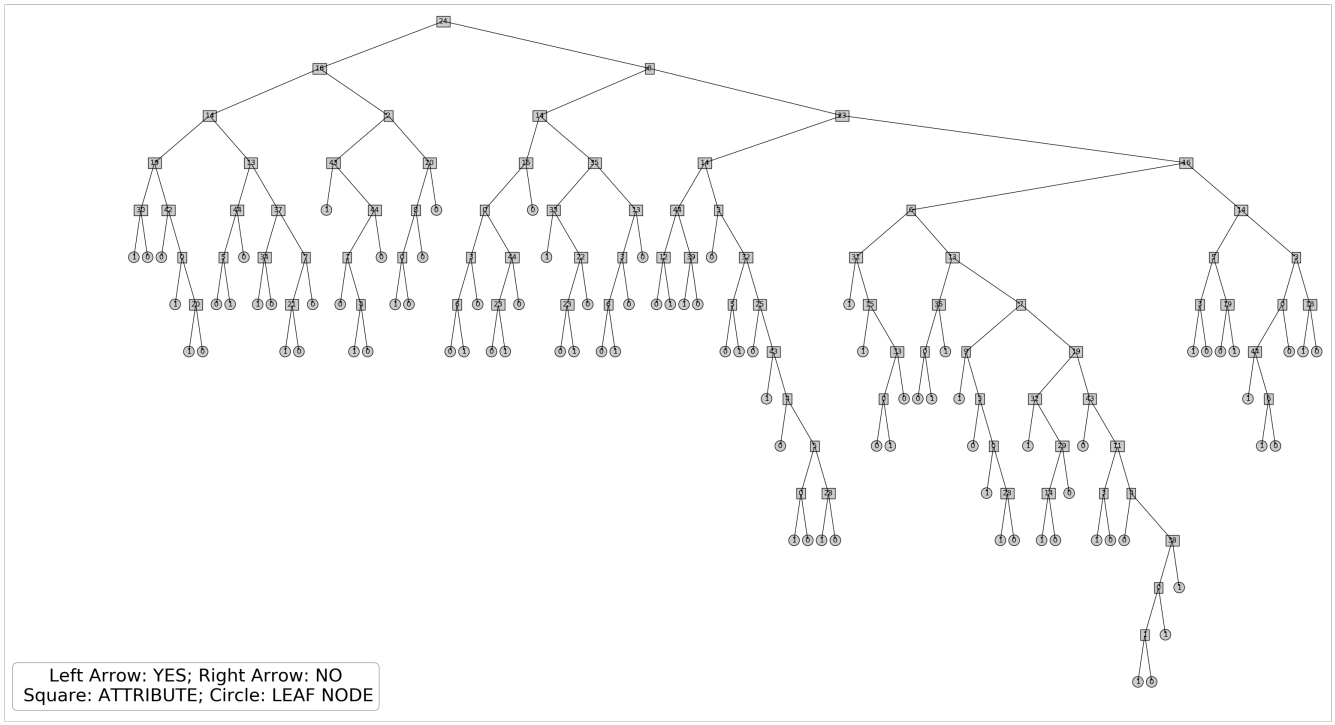

Figure 5: Decision Tree of Sadness

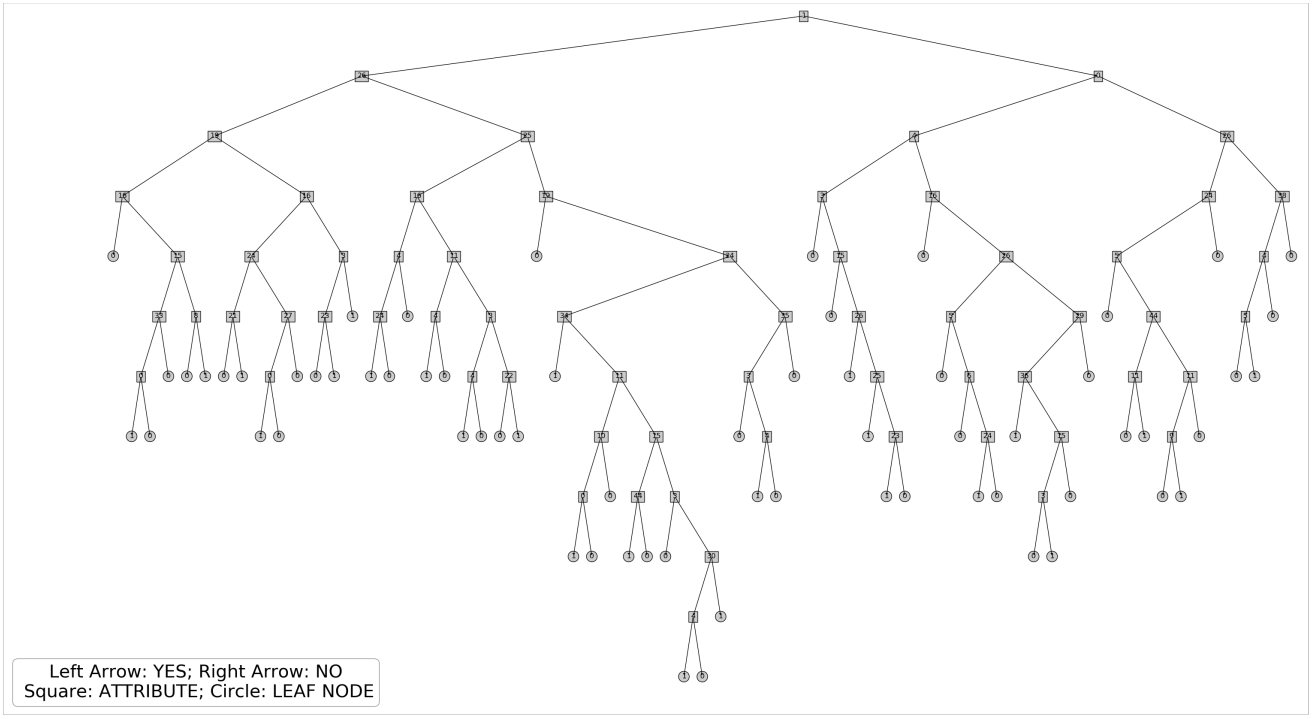

Figure 6: Decision Tree of Surprise

3.2 Confusion Matrix

Emotion	Anger	Disgust	Fear	Happiness	Sadness	Surprise	Act. Sum
Anger	8.90	1.70	0.50	0.40	1.40	0.20	13.1
Disgust	1.50	15.50	0.20	1.00	1.20	0.40	19.8
Fear	0.70	0.60	8.50	0.20	0.50	1.30	11.8
Happiness	0.10	1.00	0.20	18.40	1.20	0.60	21.5
Sadness	1.80	1.50	0.40	0.60	8.10	0.80	13.2
Surprise	0.00	0.30	1.50	0.40	0.80	17.60	20.6
Pre. Sum	13	20.6	11.3	21	13.2	20.9	

Table 1: Average Confusion Matrix for Clean Data

Emotion	Anger	Disgust	Fear	Happiness	Sadness	Surprise	Act. Sum
Anger	2.60	0.90	1.90	0.70	1.90	0.80	8.80
Disgust	1.30	12.90	1.70	1.70	0.80	0.30	18.7
Fear	1.30	1.10	11.70	1.80	1.20	1.60	18.7
Happiness	1.10	0.90	1.40	15.80	0.30	1.30	20.8
Sadness	1.90	0.80	0.70	0.70	5.70	1.20	11.0
Surprise	1.00	0.60	1.60	1.20	0.90	16.70	22.0
Pre. Sum	9.2	17.2	19	21.9	10.8	21.9	

Table 2: Average Confusion Matrix for Noisy Data

Observed from the confusion matrix, there is no significant imbalance problem within the clean test dataset, since the amounts of samples for each actual class, six emotions, do not differ remarkably.

Emotion	Anger	Fear	Happiness	Sadness	Surprise	Act. Sum	
Anger	0.295	0.102	0.216	0.08	0.216	0.091	1
Disgust	0.07	0.69	0.091	0.091	0.043	0.016	1
Fear	0.07	0.059	0.626	0.096	0.064	0.086	1
Happiness	0.053	0.043	0.067	0.76	0.014	0.062	1
Sadness	0.173	0.073	0.064	0.064	0.518	0.109	1
Surprise	0.045	0.027	0.073	0.055	0.041	0.759	1
Pre. Sum	0.706	0.994	1.137	1.146	0.896	1.123	

Table 3: Normalized Confusion Matrix for Noisy Data

However, for the noisy test dataset, the problem of imbalance becomes too serious so that it must be normalized before any further measures to eliminate the potential problems caused by the a number of imbalanced emotion samples.

3.3 Recall Rates, Precision Rates and F1 Measure

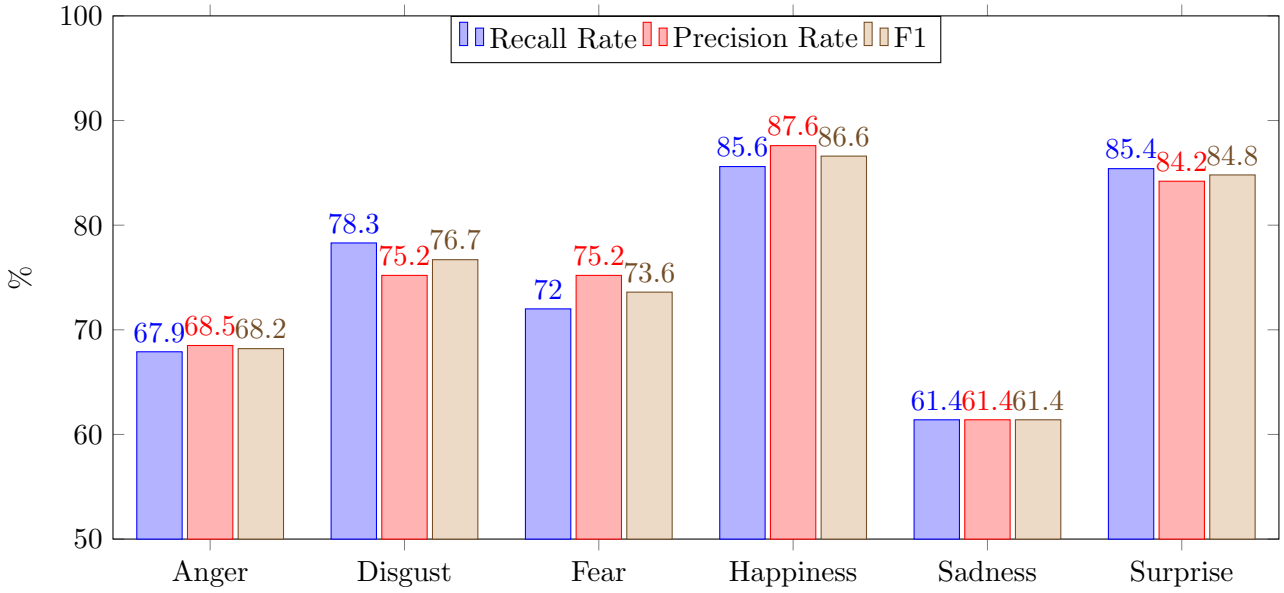

Figure 7: Bar Graph of Indicators of Clean Dataset

Focused on the graph of clean dataset, the distribution of each type of rate for six emotions can be recognized as balanced. Specifically, the happiness tree[6] achieves the best performance on all aspects(precision: 87.6%; recall: 85.6%; F1: 86.6%), while the sadness one performs worst(precision: 61.4%; recall: 61.4%; F1: 61.4%). Besides, the difference between recall rate and precision rate, for all emotions, is not significant, which means all decision trees achieve relatively equal performance on predicting positive and negative samples.

On the contrast, the performances of different emotion trees vary dramatically from the noisy test dataset with the best one, similar as the results of clean test dataset, happiness tree(precision: 66.4%; recall: 76.0%; F1: 70.8%) and the worst one, anger tree(precision: 41.9%; recall: 29.5%; F1: 34.6%). Moreover, the value of recall rate and precision rate significantly differs too, especially for the anger tree, which means such trees can only work well on one single aspect, either predicting positive samples or predicting negative samples, of prediction tasks. For example, the anger decision tree, in this case, predicts lots of positive samples wrongly but those predicted as positive are indeed positive.

[5]Matplotlib: https://matplotlib.org/

[6]the term 'tree' mentioned here can not be recognized equally as the corresponding trained decision tree. Because the statistics are calculated after combining the predictions of six emotion decision trees

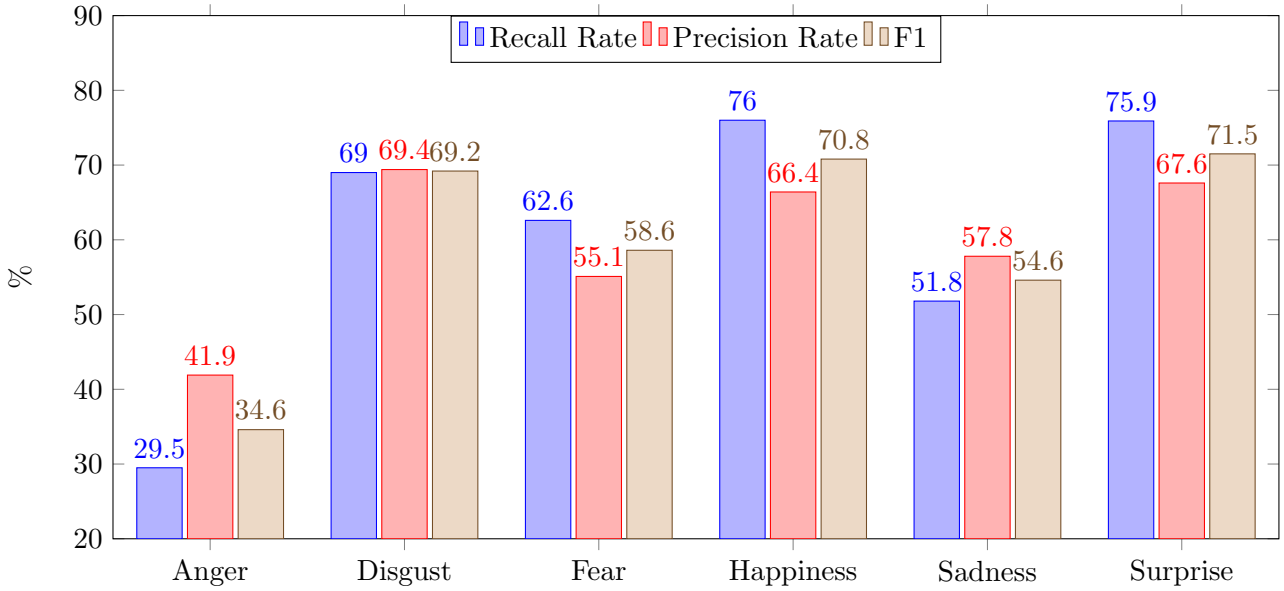

Figure 8: Bar graph of Indicators of noisy dataset

3.4 Classification Rate

Dataset	Clean	Noisy
Classification Rate	0.770	0.608

Table 4: Classification rate of two datasets

The classification rate represents the overall performance of the whole classification system containing six trained decision trees. To be more specifically, the classification rate 77.0%, taking the clean dataset as example, means 77.0% of all samples are predicted successfully including both positive and negative cases. Essentially, the value of classification rate can be, in this case, affected by the accuracy of internal individual decision tree and the effect of integration for the predictions of contained trees. Obviously, the system performs better on the clean dataset with the classification rate of 77.0%, whereas the same model but trained on the noisy dataset can only achieve a classification rate of 60.8%.

4 Question Answers

4.1 Clean Datasets v.s. Noisy Datasets

In this case, two datasets, clean and noisy, are supplied for learning and testing models. The clean dataset has been filtered by the human experts, whereas the noisy dataset is obtained by an artificial system for AU recognition[1]. Intuitively, the clean dataset has a higher accuracy on the labeling compared to the noisy dataset due to the possibility of wrongly detected or missing AUs for the noisy one.

As discussed before(3.2 - 3.4), the noisy system, referring the decision trees system built on the noisy dataset, has a worse performance, concerning both overall system and individual trees, than those of the clean system. To be more specific, for overall performance, the clean system achieves a accuracy of 77%, while the noisy one only achieves that of 60.8%. And in terms of individual emotion performance, the comprehensive performance, i.e. F1 measure, are close for two systems concerning emotions Disgust and Sadness, while the values of F1 measure differ a lot, about 10% for Fear, Happiness and Surprise and specially 34.4% for Anger. Further, the remarkable difference between the precision rate and the recall rate can be easily observed within emotions Anger(12.4%), Fear(7.5%),

Happiness(9.6%), Sadness(6%) and Surprise(8.3%) for the noisy system, whereas the aforementioned difference can be approximately ignored for the results of clean dataset.

4.1.1 Analysis and Assumptions

While training noisy data, it may be the case where two contradictory expressions have similar AUs which makes the model itself has difficulty while identifying these contradictory expression[2]. For instance, the expressions Anger and Happiness may have similar AU combinations in noisy data which leads to the case that the model cannot decide the distinct expression whether it is Anger or Happiness from one AU set. As can be seen from figure 2, the Anger result suffers the issue with low recall rate but high precision rate; this indicates the situation where there are low false negatives and high false positives. On the other hand, Fear, Happiness and Surprise have high recall rate and low precision rate which represent a lot of false negatives are treated as false positives. Both situations are more or less due to the mislabeling of noise data; there are two conditions, firstly, the model is trained with correct AU combinations but tested with wrong combinations. Secondly, the model could be trained with noisy training dataset, but tested with clean testing dataset. The other issue with figure 2 is imbalance, which means the amount of dataset provided for each expression is not equal; this will lead to the inaccurate result due to the lack of samples.

There is another assumption can be raised that with two similar expressions in real world scenario, they cannot be easily distinguished by limited number of AUs. For example, both expressions Anger and Sadness have low recognition rate in clean and noisy datasets indicate that they may have very similar combinations of AUs; in reality, these two expressions can both lead to some extent of depression.

4.1.2 Extensions

To deal with these issues, Brodley and Friedl has illustrated a general algorithm which creates classifiers with voting and consensus mechanisms in order to filter out dataset before training[3]. This method is a fairly straightforward approach to deal with mislabeled training data; another approach has been implemented within the training process, cross validation. This coursework uses 10-fold cross validation to split datasets into sub-groups to reduce the lack of randomness if picked by humans.

On the other hand, noisy data does not always lead to low performance; in CNN, it has been approved that, if the added noise is random noise, it may help enhance the accuracy[4]. In addition, it depends on circumstances whether the model is at the overfitting or underfitting situation; if it is at underfitting, the model has not fully learned the principle of this dataset or algorithm behind the scenes, hence disturb labels will affect the performance as the model may not be able to identify disturb labels; however, at overfitting, the model has understood the principle and the previous errors have been treated as part of the principle, so that new disturb labels will bring the benefits to overfitting[5].

4.2 Ambiguity

Basically, each decision tree can only produce one single boolean value, 1(YES) and 0(NO), for the prediction of its own represented(trained) emotion, which means a vector of 1*6 will be produced as a prediction of one single sample, a vector of 1*45. But, sometimes the conflicts will happen when the prediction cannot be promised as only one consistent result due to the potential situations, like none or more than one decision trees are activated.

The basic idea to solve this problem is adding a procedure of combining the results of six trees into single one output within the function *testTrees()* specifically between the predictions and output.

4.2.1 Random Picking

To use this method, the predicted emotion will be picked randomly within the range of emotions represented by the activated, i.e. returning YES, decision trees or all six emotions if no decision tree predicting YES.

This method can be taken as the baseline for any other combining ways due to its unique randomness, which means nothing artificial involved in the picking. The drawback is, also caused by the property of randomness, the fluctuation of prediction results. Therefore, to solve this problem, all performances mentioned below related to this method will take the average value of 5 times of calculations.

4.2.2 Performance Picking

Intuitively, the decision tree with greatest possibility to predict successfully should be selected if more than one decision trees activated, while the one most possibly to mispredict, predicting NO wrongly, should be picked up under the situation none activated. Both of two aforementioned possibility are corresponding to the precision rate and recall rate respectively. In other words, for the first situation, the tree with highest precision rate will win and for the second situation, the winner will be the one with lowest recall rate.

Advantages The logic behind this method is very easy to understand and consistent to our intuition. Moreover, it should have an positive effect on improving the performance, i.e. accuracy.

Disadvantages The performance of this method significantly depends on the validity of the assumption that the future unseen testing dataset is also drawn from the same distribution, which means the accuracy will be decreased as the difference between two distribution largens. Besides, the accuracy of involved measure indices can also affect the results. To be more specific, the inaccurate measurement is expected to lead a wrong picking on the activated decision trees. Eventually, obtaining such indices is built on the sacrifice of a part of data as a validation set to calculate them, which is supposed to slightly reduce the final performance too.

4.2.3 IG Picking

Considering each prediction as a decision making, the accuracy of decision will be heavily depended on the information concerned during the procedure. Thinking in this aspect, the previous method Performance Picking only takes into account the general information of the whole decision tree assuming that the possibility of each particular prediction path, from the root node to the leaf, is equal. In fact, even for the same one decision tree, the reliability of each prediction path can be dramatically different due to the various amounts of samples used to generate the node or leaf during training. Therefore, this method will consider the solution based on the accumulated information gain, calculated during training to determine the best attribute, as a measure of how much uncertainty has been eliminated on the specific prediction path. For example, the accumulated IG for one specific prediction path(A-B-YES), A(root; IG=1) - B(IG=2) - YES(prediction), is 3(1+2). Basically, higher accumulated IG means that more uncertainty is reduced within the data, which in other words the result is more certain.

Advantages IG Picking specifies the decision scope for each prediction to the specific single one path, how the prediction was made, from the overall performance of whole decision tree. So the priority for picking will change for each prediction and each decision tree due to the different activated prediction path, not like the Performance Picking fixed as the order of precision/recall rate values.

Disadvantages Similarly, the performance of this method is still restricted by the distribution of future test data. Besides, the accumulated IG is directly related to the uncertainty elimination, i.e. the states of attributes and their distributions. Thus it can be only taken as a approximate substitution of the possibility that the prediction is right.

4.2.4 Comparison

The results of experiments have been shown as expected: IG Picking(77.0%) > Performance Picking(73.1%) > Random Picking(72.0%). Moreover, concerning the separated performance on the two specific conditions, the strategy, the best tree for more than one activated and the worst tree for none activated, also achieves balanced performance on such two conditions.

4.3 Pruning Example

The *pruning_example* function works in the following steps:

1. use *classregtree()* function to create a full classification decision tree without pruning.
2. use 10-fold cross validation to compute the cost vector, standard error vector, the number of terminal nodes of each subtree, and the estimated best level of pruning. Then compute the above four parameters using the resubstitution method, which uses training data for validation.
3. confirm the different best levels for pruning calculated from the second step.
4. find the minimal costs of the two cost vectors and their corresponding indexes.
5. plot the relationship between the cost and the number of terminal nodes, and then according to the estimated best level of pruning, find the corresponding cost and the tree size.

4.3.1 Diagrams Description

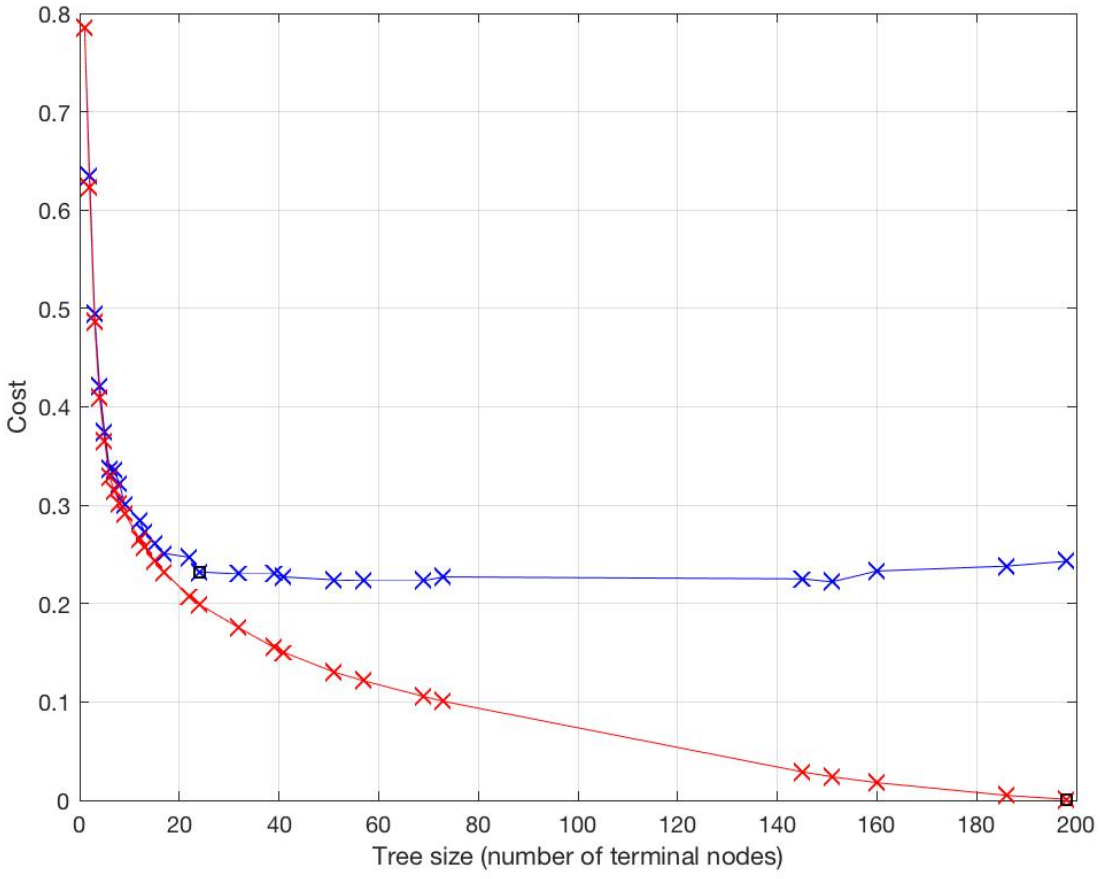

Figure 9: Clean Data

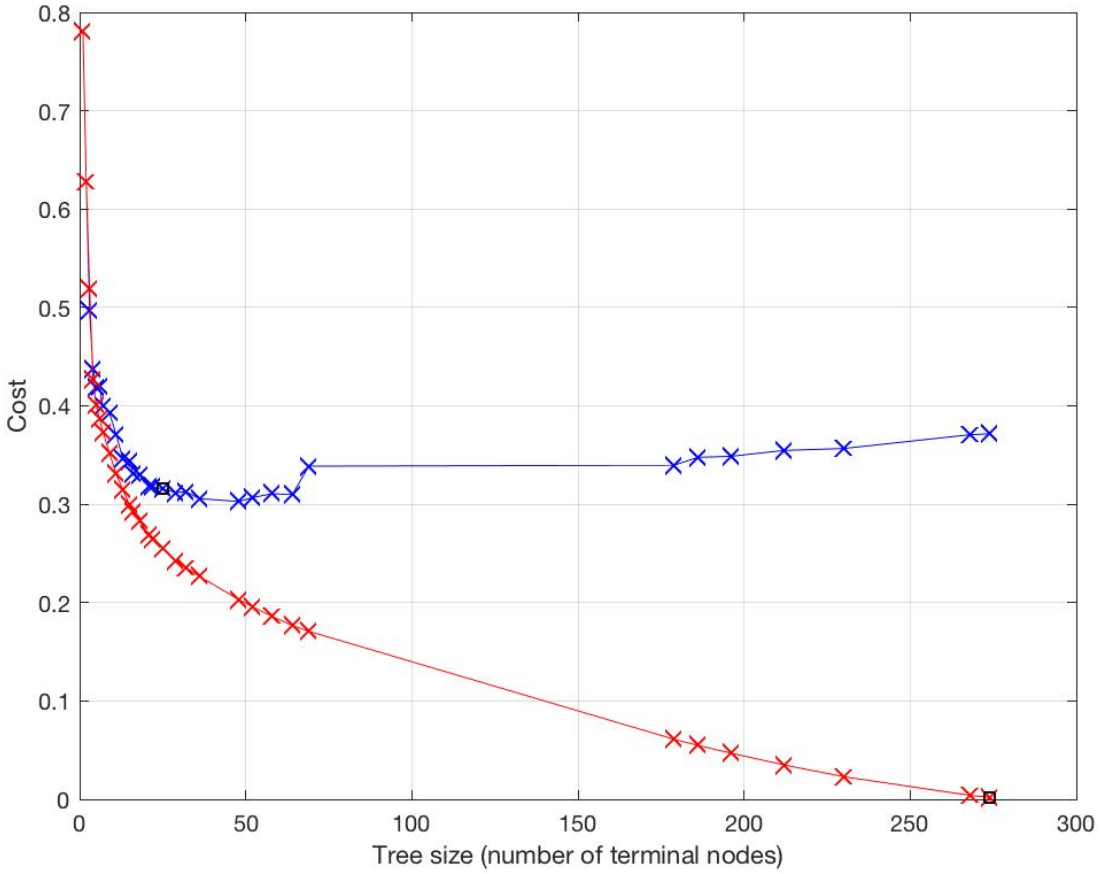

Figure 10: Noisy Data

In each figure, both curves represent the change of cost with respect to the number of terminal nodes. The blue curve refers to the results generated by the 10-fold cross-validation, and the red curve refers to the resubstitution method.

In both figures, with the increase of the number of terminal nodes, the red curve always decreases and ends up with cost being 0. Because resubstitution method uses training samples for validation, and also suggests that the generated tree overfits the training samples. The blue curve decreases as the tree size grows, and then either plateaus, which occurs on the clean data, or at a certain point starting to increase, which is obvious on the noisy data. The reason is that the performance of the decision tree on a new dataset will be not improve or even degraded when the tree overfits the training samples.

4.3.2 Difference Analysis

The red curve of clean data decreases faster than that of noisy data, and the tree size of clean data is smaller than that of noisy data when the red curves reach the point of cost being 0. The blue curve in both figures decreases first, but on clean data, it then starts to plateau, which means the growth of tree size no longer has significant impact on the decrease of cost, while on noisy data, it increases sharply at a certain point, which indicates continuing to increase the tree size will start to make the performance worse.

4.3.3 Optimal Tree Size

The threshold of optimal tree size for each case is the point when the problem overfitting is going to matter, i.e. the difference between the red and blue curves reaches the specific range. For the clean

dataset, the optimal tree size is approximately 24, and for the noisy dataset, the optimal tree size is around 25. In general, if tree size is too small, then the classification error will be high, while if the tree size is too large, then the model will be overly complex, and its predictive power on new data set will be reduced.

References

[1] Michel F Valstar and Maja Pantic. Fully automatic recognition of the temporal phases of facial actions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(1):28–43, 2012.

[2] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review*, 22(3):177–210, 2004.

[3] Carla E Brodley and Mark A Friedl. Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167, 1999.

[4] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. Disturblabel: Regularizing cnn on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4753–4762, 2016.

[5] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.