**Performance of Interval-Valued Dataset in managing uncertainty in Crowdsourcing against the Inference Algorithms**

**Introduction**

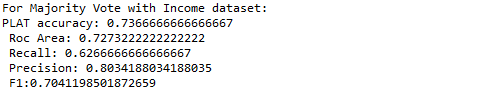
The quality of data that is extracted from the users in crowdsourcing can vary due to its open nature and varied level of human expertise. Therefore, there are various ways to handle this uncertainty in the labeling of the data where one can use the ground truth inferences algorithms along with the learning models and try to overcome the issues. The other way around is making use of interval-values in which each user provides the probablistic range in which they believe that this instance belongs to a certain label. Then with the help of various statistical and probablistic approaches this IVLs are pre-processed and handled to make a final conclusion about the label values.

**Methodology used**

Initially the ground truth inference algorithms were executed and we had compared their performances on the Income dataset. The dataset contained 4 files which were .gold.txt which contained the ground truth of each instance (Instance ID and True Label). The .reponse.txt file which defines the true label value obtained for a instance from a worker (Worked ID Instance ID Label). The .arff file which provides the information about features and labels of each instance. The .arffx file which provides information about the worker which has provided the information.

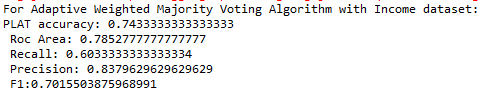
All these files and operations were performed with the help of CEKA tool which helps to mine the wisdom of crowds. This dataset was then further used to implement different ground truth inference algorithms. They are as follows:

1. **Majority Voting Algorithm:** Most simples inference algorithm. Gives the integrated label a label value based on the condition that class has maximum labels. If there are same number then class is randomly choosen.

****

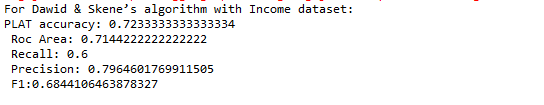
**Fig. 1. Result of MV Algorithm**

1. **Adaptive Weighted Majority Voting Algorithm:** It is developed to handle imbalanced labeling issue. Here initially positive and negative class have weight set to 0.5 and are then adjusted to get exact weight such that sum is 1.

****

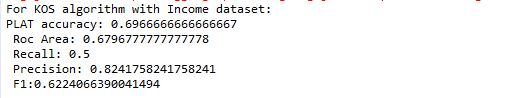
**Fig. 2. Result of Adaptive Weighted Majority Voting Algorithm**

1. **Dawid & Skene’s Algorithm :** It uses confusion matrix where each confusion matrix presents a different worker where elements of it depict the probability that the class label k is correct for the current label.

****

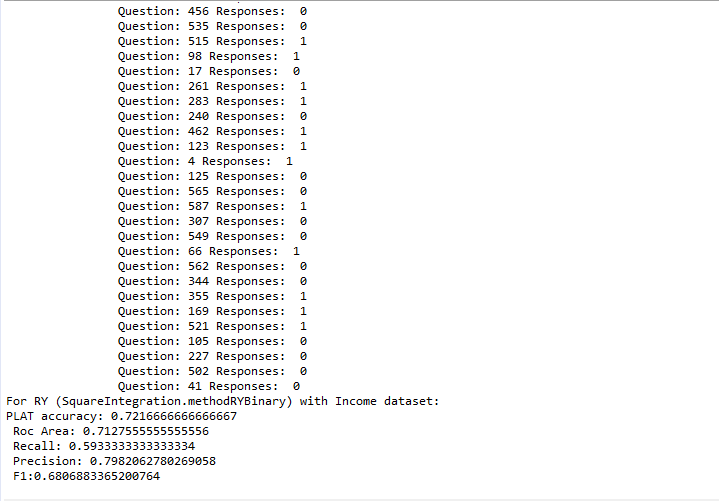
**Fig. 3. Result of DS Algorithm**

1. **KOS Algorithm:** It was motivated by the reality of differences in labeling quality among different labelers. The authors sought to infer what they call the “correct answers” to “tasks,” analogous to the ground-truth labels of examples.

****

**Fig. 4. Result of KOS Algorithm**

1. **RY Algorithm:** RY uses baseian approach to model the probablities to evaluate the bias towards the positive class (sensitivity) or the bias towards the negative class (specificity).

****

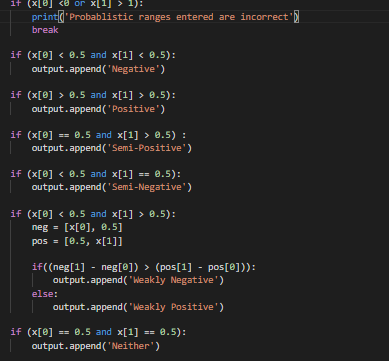
**Fig. 5. Result of RY Algorithm**

From above we can see that the KOS Algorithm performes the worst and the adaptive weighted majority voting algorithm performes the best. The issue with these algorithms is that it wants users to either completely accept or reject that a particular instance belongs to a certain class whereas in certain scenarios even the experts of that field may not be sure.

Here comes into the picture the interval valued dataset which allows users to provide a range within which they accept or reject a particular instance being of a particular class. This helps to take into account the user uncertainty. So the user provides a low and high range value for the same which is within the range of [0,1]. Therefore, an IVL of [0.8,0.9] means user accepts that a particular instance belongs to a class with a 80-90 percentage of certainty.

This approach helps us to take into account the user uncertainty as it depicts us that user is certain about an instance for a particular probablity and also shows that user believes that the instance can be dismissed of being belonging to a particular class.

An IVL can be classified as either Positive (+), Negative(-) or neither of the above. If both the lower and upper limit are greater than 0.5 than its labeled as Positive. If both the lower and upper limit are less than 0.5 than its labeled as Negative. If both the limits are 0.5 than it belongs to the neither category as it means that the user is undecided. There are certain cases where the lower limit can be 0.5 and upper limit can be greater than 0.5 this means that user is not absolutely certain about it being positive thereby it can be labeled as Semi-Positive. If the lower limit is less than 0.5 but the upper limit is 0.5 which means the user is not even decided whether the instance is negative thereby it is labeled as Semi-Negative.This issue can be further also solved by using the Weakly-Positive and Weakly-Negative concept. In this case we split the limit into two one which contains the lower limit and 0.5 and another which contain 0.5 and upper limit. Then we calculate their differences and then if the negative difference is higher its labeled as weakly-negative and if not then its labeled as weakly-positive.



**Fig. 6. Conditions for setting the instance class output**

The user input for the dataset is as follows:



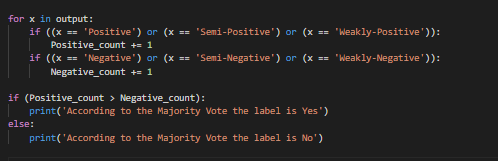
**Fig. 7. User Input Data**

This data is then parsed through the conditions mentioned above and then the output is generated which classifies each of the label limits into one of the categories:



**Fig. 8. Classified Output**

This output is then parsed to the Majority Voting algorithm which basically checks which class dominates over the other in-terms of count. That class is then voted as majority.



**Fig. 9. Majority Voting**



**Fig. 10. Majority Voting Output**

Before the data is parsed to this it is pre-processed to remove the incorrect ranges of data and to handle the Semi-Positive and Semi-Negative cases.

This is a very simple and easy approach to implement where it just takes into account whether the label is positive or negative at the end and counts them. This approach only considers if an IVL is positive or negative but does not take into account its values. Hence, an inference with MV may not have high probablity of matching.

**Main Approach:**

With the method above we are able to make an inference where we are able to classify an instance xi to a label Li. The approach implemented now also provides an information in terms of what is the probablity with which an inference matches the ground truth. For this we take into account the concept of probablity density function. We need to calculate the probablity density function of each instance and label and then use that further to calculate the probablity of matching.

A function f(t) is called probablity density function of a set of IVL labels if it satisifes following:

f(t) >= 0, ∀t ∈[0,1]; and

Σks=1 ∫0, lij f(t)dt = 1

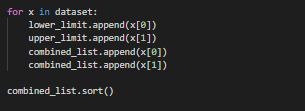
The pdf can make use of any of the known distributions for the example demonstrated below I have made use of the uniform distribution. The Pdf can be finally aggregated and calculated as follows:

F(t) = (Σks=1 pdfs (t)) / k



**Fig. 11. Input for the algorithm**

The first step according to algorithm is to split up the upper and lower limits and concatenate them into a sorted list

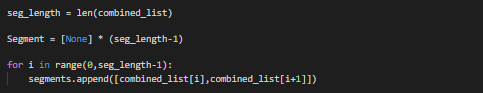


**Fig. 12. Combined Sorted List**



**Fig. 13. Combined Sorted List Output**

Create the segments of intervals from this combined list according to the algorithm

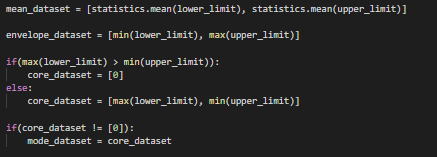


**Fig. 14. Segments**



**Fig. 15. Segments Output**

Statistical information about the data



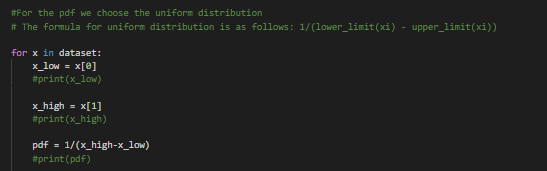
**Fig. 16. Statistical Information**



**Fig. 17. Statistical Information Output**

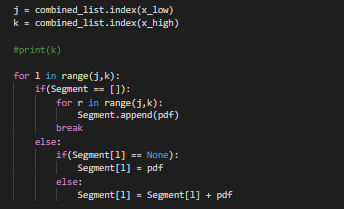
The distribution used for this example is the uniform distribution. The pdf of uniform distribution is =

1/(B-A) where B is the upper boundary and A is the lower boundary.



**Fig. 18. Distribution used**

Next step according to algorithm is to accumulate the pdf for each segment which is done by taking each data into consideration we find the values of j and k and then run a loop to find the pdf for each segment.

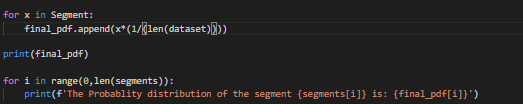


**Fig. 19. Accumulate the pdf on each segment**

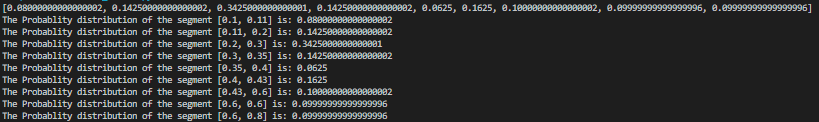


**Fig. 20. Accumulated pdf for each segment**

Now calculate the pdf iterate over the segment and calculate pdf for each segment

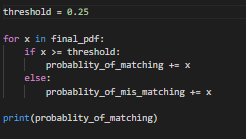


**Fig. 21. Pdf for each segment**



**Fig. 22. Pdf for each segment output**

Now to calculate the probablity we need to set a threshold of the value whose pdf should be added into the category of being accepted. I have set that as 0.25. So anything from 0.25 to 1 their pdf will be added to get the final probablity of matching and one minus that value will give the probablity of mismatching.



**Fig. 23. Probablity calculation**



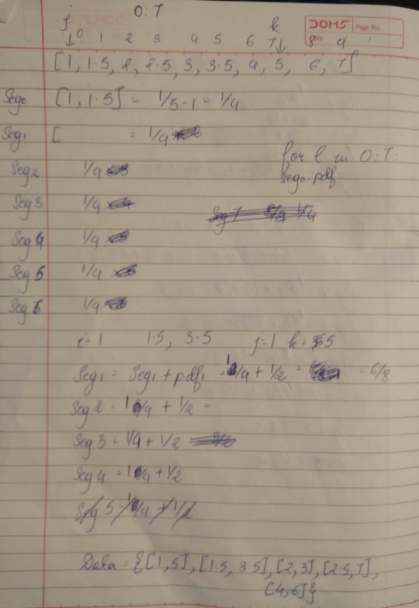
**Fig. 24. Probablity of matching**

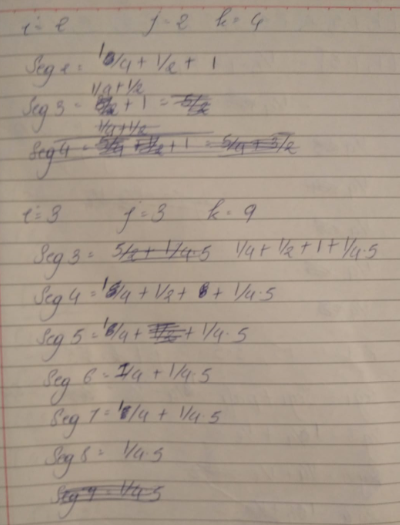
Thus, this way using a IVL dataset we calculated the probablity which is shown as above. The quality of ouput probablity is currently low which is due to the input that is supplied as above which can be improved further. This can be done by filtering out low quality labels and IVLs and performing the pre-processing step thoroughly.

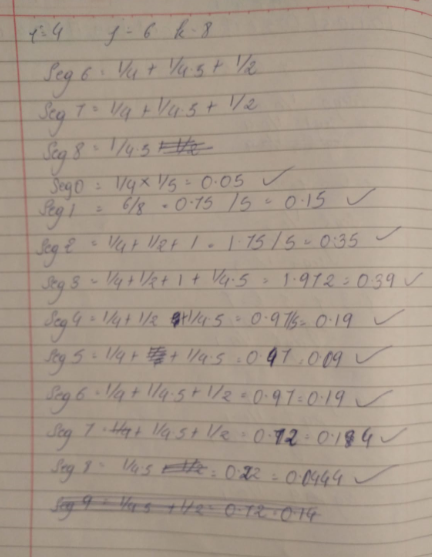
Thereby, here I have implemented the IVL along with the Majority voting as well as the probablity calculation of matching has been done so as to check the overall quality of labels. We are able to infer the probablity of matching on the basis of threshold which gives us the quality of our labels.

Below is an example of pdf calculation using uniform distribution:

Data = {[1,5], [1.5,3.5], [2,3], [2.5,7], [4,6]}







**Github Repository :**