**CEKA Execution**

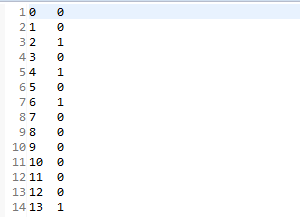
**Steps for CEKA Installation:**

1. Download Eclipse
2. From the Eclipse Marketplace install Subclipse 4.3.0
3. Go to file -> Import -> SVN -> Checkout Projects with SVN -> New Repository -> Add this given URL (<https://svn.code.sf.net/p/ceka/code-0/branches/release1.0>) -> Name the project -> Finish
4. Go to project name -> Right Click -> Properties -> Build Path -> Add Ceka/Test
5. Go to jar -> Add Jar -> Add all the jar files from lib with .jar extension
6. Go to file -> Import -> SVN -> Checkout Projects with SVN -> New Repository -> Add this given URL (https://svn.code.sf.net/p/ceka/code-0/tags/Weka-3.6.10) -> Finish
7. Go to project name -> Right Click -> Properties -> Build Path -> Libraries -> Remove weka.jar
8. Go to Projects in the same window and add Weka to it
9. Create your own new project in the properties go to project add both ceka and weka to it. In the libraries add all the jar files from lib with .jar extension.

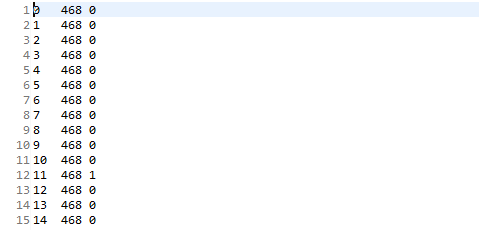
**Datasets:**

**There are 4 types of input files that we use here:**

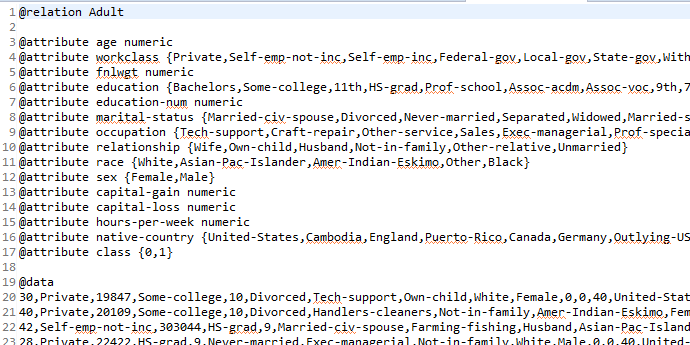
**.gold.txt :** Contains the ground truth of each instance in the dataset. Format: Instance ID True Label. Ex : Income94.gold.txt

****

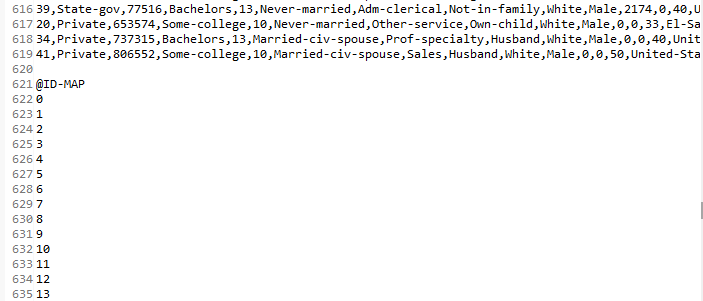
**.response.txt :** defines true label value obtained for a instance from a worker. Format: Worker ID Instance ID Label. Ex: Income94.response.txt

****

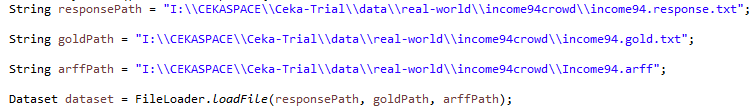
**.arff :** provides information about features and labels of each instance. Ex: Income94.arff

****

**.arffx:** an extension to .arff file in weka to provide information about the worker which has provided the information. There are two parts one for data and @id-map for worker id.

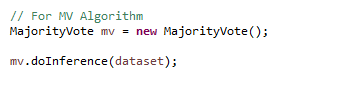
****

**These files are now combined together into a dataset using the dataset class as follows:**

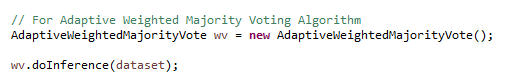
****

**Algorithms:**

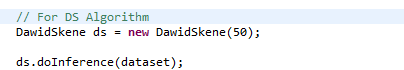
1. **MV Algorithm :** Most simples inference algorithm. Gives the integrated label a label value based on the condition that class has maximum labels. If there are same number then class is randomly choosen.

****

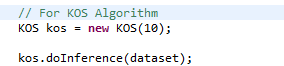
1. **Adaptive Weighted Majority Voting Algorithm:** It is developed to handle imbalanced labeling issue. Here initially positive and negative class have weight set to 0.5 and are then adjusted to get exact weight such that sum is 1.

****

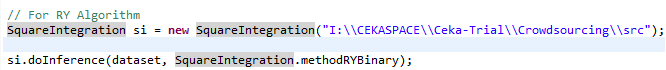
1. **Dawid & Skene’s Algorithm :** It uses confusion matrix where each confusion matrix presents a different worker where elements of it depict the probability that the class label k is correct for the current label.

****

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

****

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).

****

**Noise Filtering Algorithms:**

So the data obtained via application of inference algorithms has still instances where labels are incorrect which result in reduction of accuracy and other paramaters that detect the performance. Therefore in this step we identify the set of instances which have noisy(mislabeled data) and correct them.

**filternoise():**

This function is responsible for separation of the noisy data from the cleansed data it identifies the data which is highly probable to be noisy and stores it in the dataset of noisy data and stores the clean data in cleansed data dataset.

It takes in two parameters the dataset itself and the list of classifiers that can be applied to extract the mislabled data.

The function to extract the clean and noisy data are **getCleansedData()** and **getNoiseData()** respectively.

Then the ClassificationFilter class is implemented with the paramater of number of folds parsed to it. It splits the data with these fold value and then applies the learning algorithm to rest of the data and then this learnt data is results into classifiers. These classifiers are then labeled as either correct or mislabeled by comparing them to the value assigned by the trained classifier. The ones with the mislabeled values are added to the noise dataset and one with correct values are added to the cleansed dataset.

The other options to the ClassificationFilter are Majority Voting Filter, Iterative Partitioning Filter and the Multiple Partitioning Filter.

****

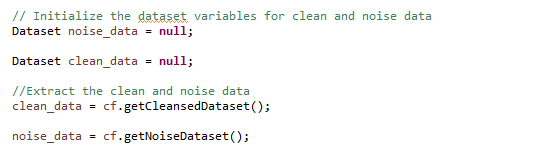
ClassificationFilter class is called and is assigned an object name of noiseFilter and then the fold value is set to 10.

****

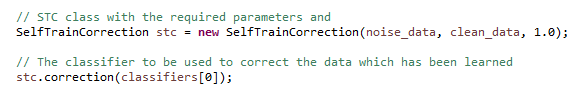
The classifier parsed to the list here is **SMO** classifier (Sequential minimal optimization algorithm) which is used in SVM to train it on the classification tasks on datasets.

Now call the filter noise function and parse the dataset along with the classifier list. Then extract the noise data and the cleansed data.

****

****

Once we have separated both the data next step is to clean the noise data and make it as accurate as possible. For that we use the **Self-Training Correction Algorithm**. We use the **SelfTrainCorrection** class and to it we parse the clean data, noise data and the proportion of data that has to be cleaned in the noise data. And we also use the correction function in order to notify that once the machine trains on the learning data the classifier it has to use to validate the incorrect labels of noise data against and correct them.

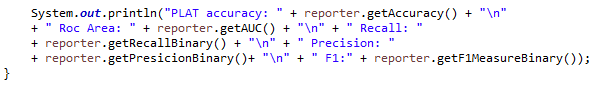
****

Once labels have been cleant and corrected we can then merge both the data then inorder to evaluate the performance of the algorithm which has been used. To merge the data:

****

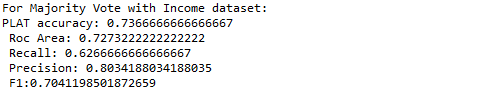
Now that the data has been merged with the correct labels we can evaluate the performance of the inference algorithm that we had used for the dataset. For that PerformanceStatistic class can be used on our dataset to obtain the statistical data about the performance of inference algorithm used. The final results obtained here are Accuracy, AUC, Precission, Recall, F-score.

****

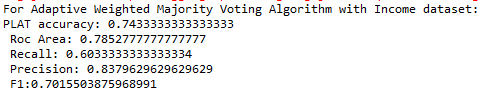
****

**Ouput:**

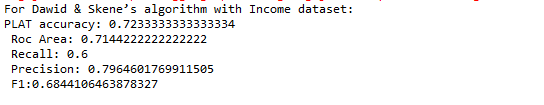
1. **MV Algorithm:**

****

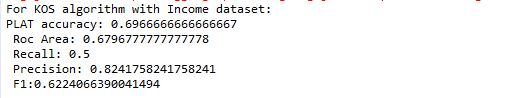
1. **Average Weighted MV Algorithm: (Best Performance)**

****

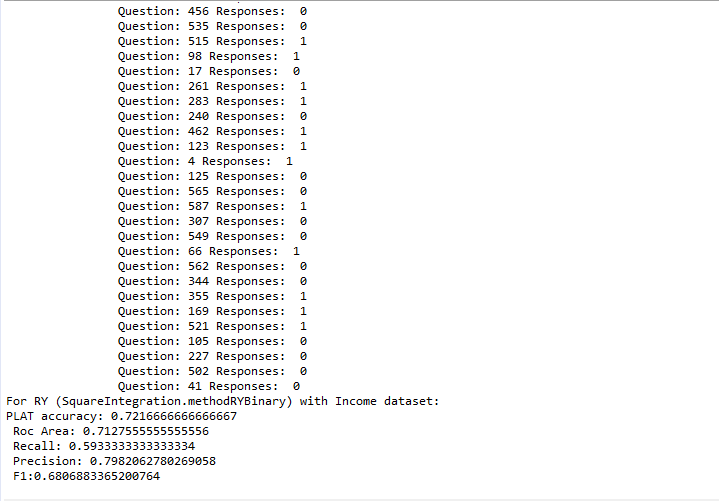
1. **DS Algorithm:**

****

1. **KOS Algorithm: (Worst Performance)**

****

1. **RY Algorithm:**

****