CSE 5243: Introduction to Data Mining

Assignment 5

Vaibhav Devekar (devekar.1)
Akshay Nikam (nikam.5)

Goal of the Assignment

This assignment aims at demonstrating classification based on association rules as well as using clustering for improving the performance of the classification

Performance Metrics

1) Accuracy by at least one matching
   This is an optimistic metric. It is based on having at least on match between predicted labels and actual labels for each test instance. We then average over the entire test instance.

2) Accuracy by one-to-one matching
   In this case, we count the number of same pairs between predicted and actual and divide it by the least of sizes of predicted and actual labels. We then average it over all test instances.

3) F-measure(Harmonic mean between Precision and Recall)

$$F - measure = \frac{2 * TP}{2 * TP + FP + FN}$$

   We take the readings for these metrics for two different cases:
   a. Fixed number of rules applied to each test instance
   b. Variable number of rules applied to each test instance until the actual number of labels is predicted

Assumptions

1) If K is the number of rules applied to each test instance, we choose K = 5, since the number of class labels for each article is 3- 5.
   Additionally, we also apply rules until we get the number of labels equal to the actual number.
2) The clusters from K-means algorithm were obtained using Cosine distance metric, since the cosine metric gave better results than Euclidean distance metric.
3) We also identify the cluster for test instances in Algorithm 2 using Cosine metric.
4) Only topics are considered as class labels. The dataset, thus, consists of 11305 records on which we perform 80-20 split.

ALGORITHM 1

The algorithm was implemented with two stages:

1) Train Phase
   After reading in the transaction matrix, we perform the following steps:
   a. Write the matrix to file to pass as input to the Apriori program.
   b. Invoke the Apriori program. It yields association rules from the given transactions. (For this step, we used an off-the-shelf tool available on KDNuggets [1])
   c. Read the rules from the output file, discarding ones which do not match the form W -> C, where W are the words while C is a class label.
   d. Order the rules in the descending order of confidence. Rules with same confidence are ordered by descending support.
   e. Subsume the rules, i.e. apply the ordered rules on the training set (transactions) in the same order. We apply rules one by one to the entire training set and remove the transactions that were covered, until all possible transactions are covered. In the end, there will be rules that were not used which we remove.
   f. Collect the transactions that were not covered by the rules and repeat the above steps for these transactions until most are covered. This will add more rules with less overall confidence, but they will help us cover most of the training transactions.

2) Test Phase
   Using the subsumed rules from above phase, we apply it on the test set. As stated before, we apply the rules twice; one with a fixed K (fixed number of rules applied to each test instance) and in other case, we keep applying rules until we get labels equal to the number of actual labels. If no rule is applicable, we apply the default label from the transactions that were not covered in train phase.

Results and Performance:

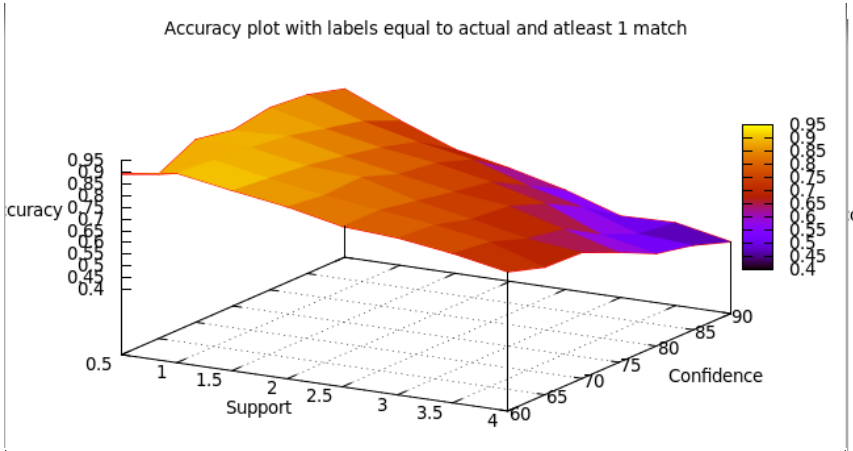
We ran the algorithm for various values of support and confidence and used accuracy, f-measure, training time and testing time as performance metrics. We tried support values in the range [0.5… 4.0] and confidence values in the range [60… 90].

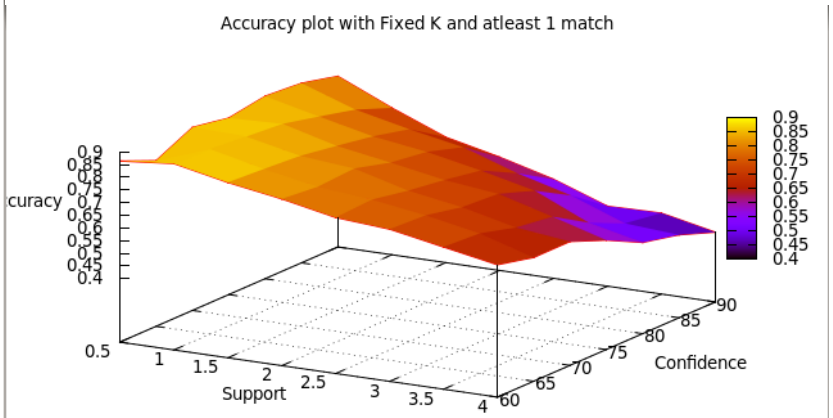
We ran two different variants of the algorithm, (1) fixed number rules being applied to a test instance and (2) as many rules being applied as required to predict the number of labels that actually exist for the test instance. Hence, we report all the performance metrics for both cases.

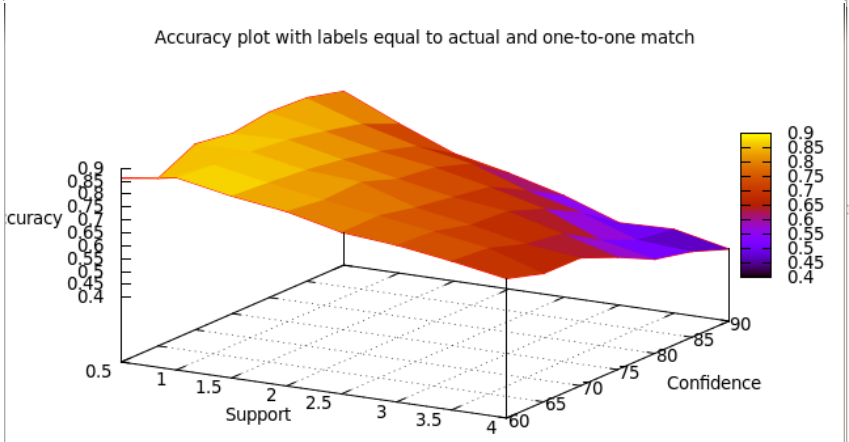
Accuracy:

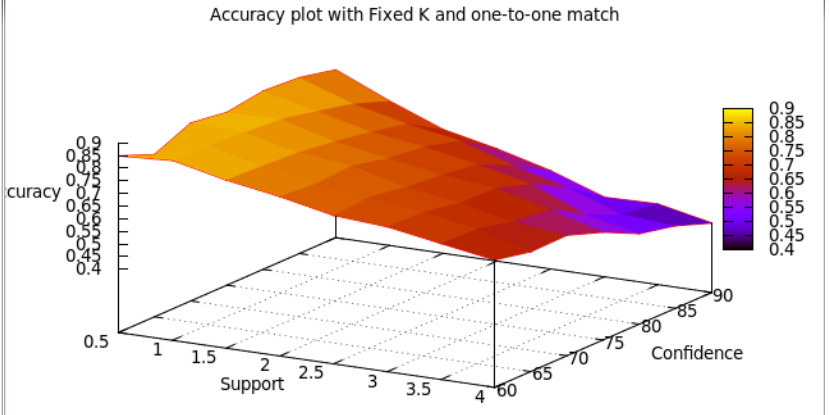
We used two different types of accuracy calculations as described in “Performance Metrics” section in the beginning of the report. Following graphs show the variation of accuracies over different values of support and confidence.

It can be observed that the accuracy value reduces with increase in support with constant confidence. With constant support, accuracy decreases with increase in confidence. Thus lowest accuracy is found at highest confidence and highest support, while the highest accuracy is found at the lowest support and lowest confidence. This is expected, since low support and confidence yield more rules that can cover most of the training and testing examples correctly.


Accuracy plot with labels equal to actual and atleast 1 match
curacy
Support
Confidence

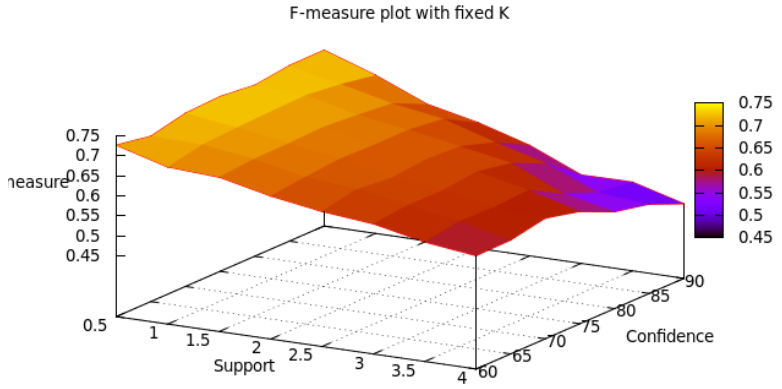

Accuracy plot with Fixed K and atleast 1 match
curacy
Support
Confidence

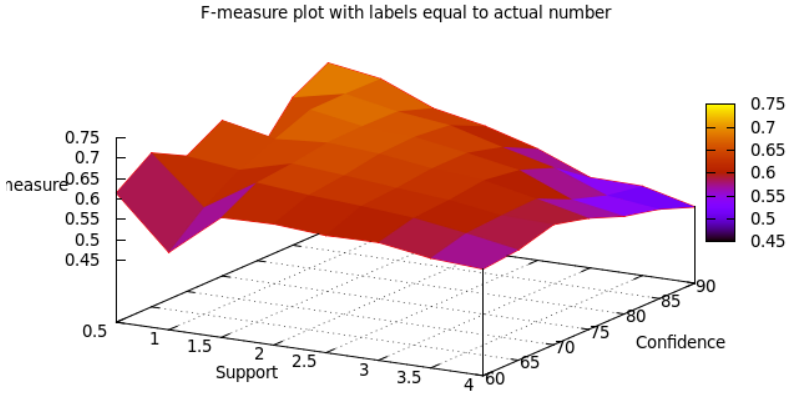

Accuracy plot with labels equal to actual and one-to-one match
curacy
Support
Confidence


Accuracy plot with Fixed K and one-to-one match
curacy
Support
Confidence

F-measure:

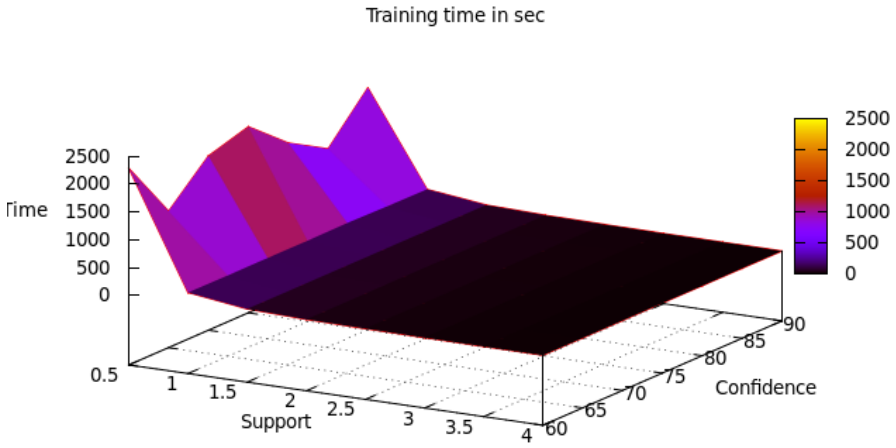
The following plots show variation of F-measure for different values of support and confidence. Again the two plots correspond to two variations of the algorithm. The trend for F-measure is same as in accuracy. High F-measure values were found at low support and low confidence values.





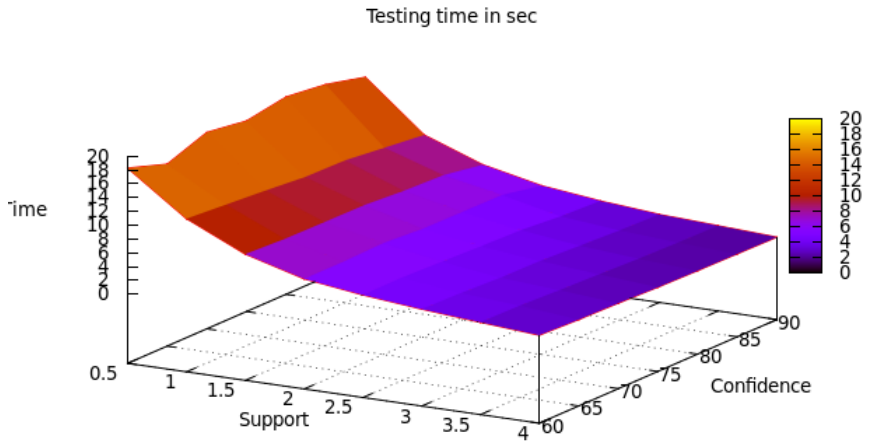
Training time:

Training time is the time required for generating association rules and subsuming them to generate a final set of rules that we apply on the test set. Following graph shows training time for different values of support and confidence. Small values of support take significantly longer because the number of rules generated is very high and it takes much longer to process them to get a final set of rules.



Testing time (time to classify):

Testing time is the time required for classifying all test instances given the association rules from training phase. As it can be seen from the following graph, the time to classify goes on reducing with increasing support. Again, since the number of rules generated is higher for low support, it takes longer to classify the test instances.



A trade-off can be observed in accuracy and running time. Higher accuracy corresponds to slower execution time, while lower accuracy to faster execution.

ALGORITHM 2

We reused the code from Algorithm 1, albeit with a few modifications. We also use the Kmeans code from Assignment 4 to obtain the clusters. The algorithm consists of following steps:

1) Clustering
   We use K means code to obtain the clusters from the training set. We obtain various number of clusters with Cosine distance with multiple parallel runs of K-means. The output is used to reconstruct transaction matrix for each cluster.
2) Train Phase
   This phase is same as that of algorithm 1. In this phase, we apply the Algorithm 1 for transaction matrix of each cluster and store the subsumed rules.
3) Test Phase
   We derive the representative means for each cluster. Using Cosine metric, we identify the cluster to which each test instance is closest to. We apply the subsumed rules from that cluster on the test instance to obtain the class labels.

Results and Performance

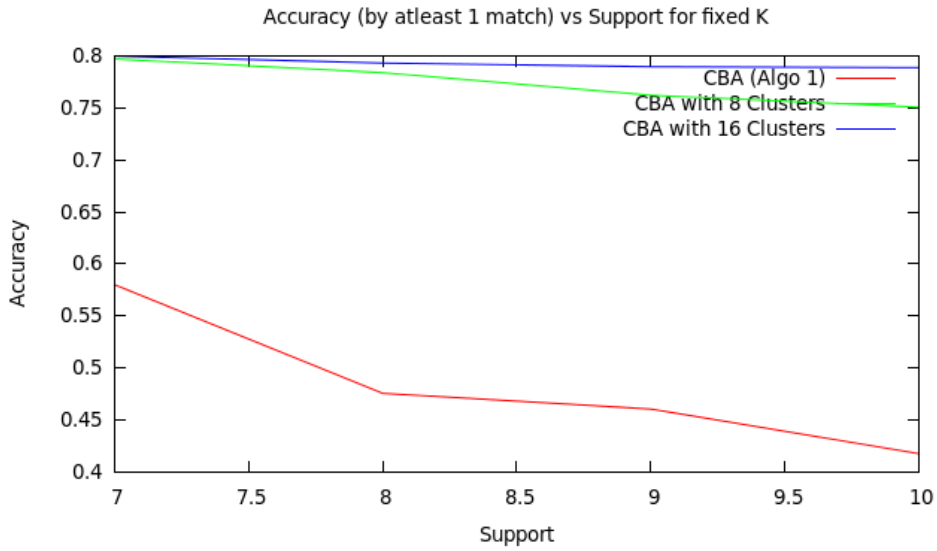
We ran the algorithm for various number of clusters starting from 8, 16 … 64. However, the apriori tool generated way too many rules for number of clusters more than 16 that could not be consumed by a python program. Hence, we show the results for number of clusters= 8 and 16 and compare them with the results from algorithm 1.

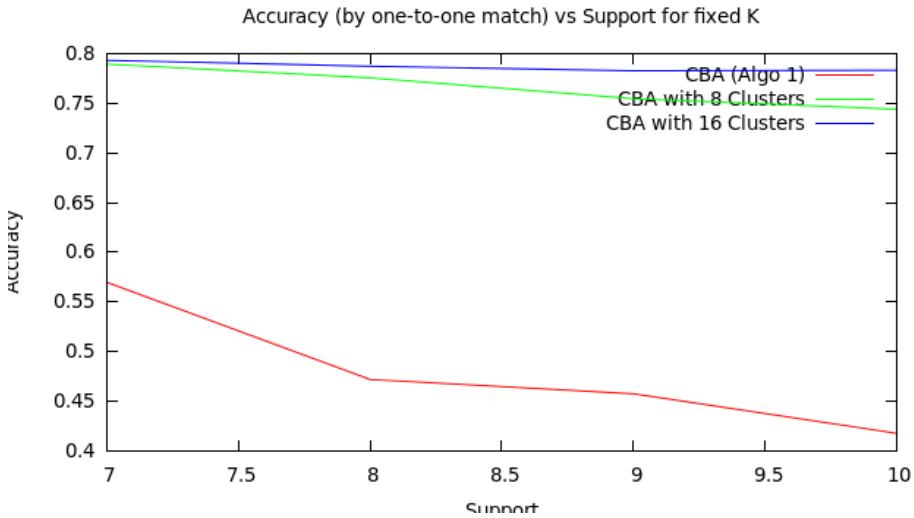
For algorithm 2, we again ran the program for various values of support (7 to 10%) and confidence (60 to 90%). For the sake of brevity, we show the results for only a variation support with fixed confidence. Following metrics were chosen to evaluate performance.

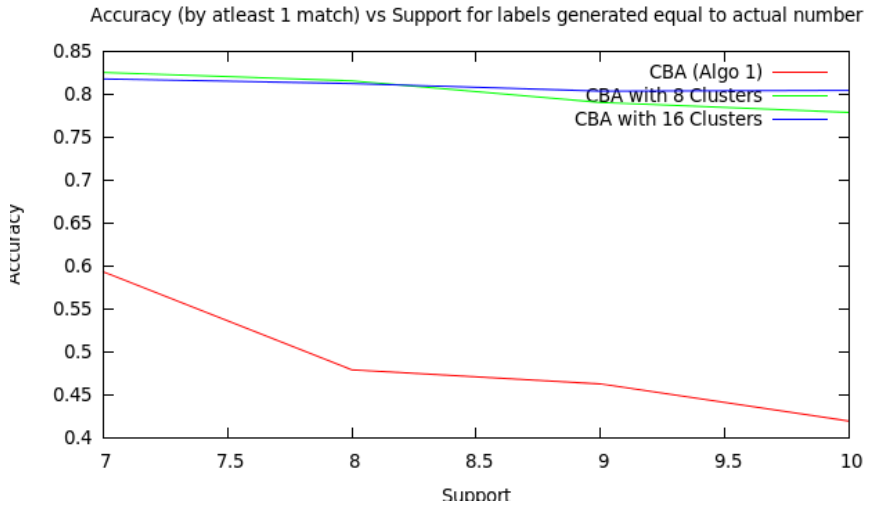
Accuracy

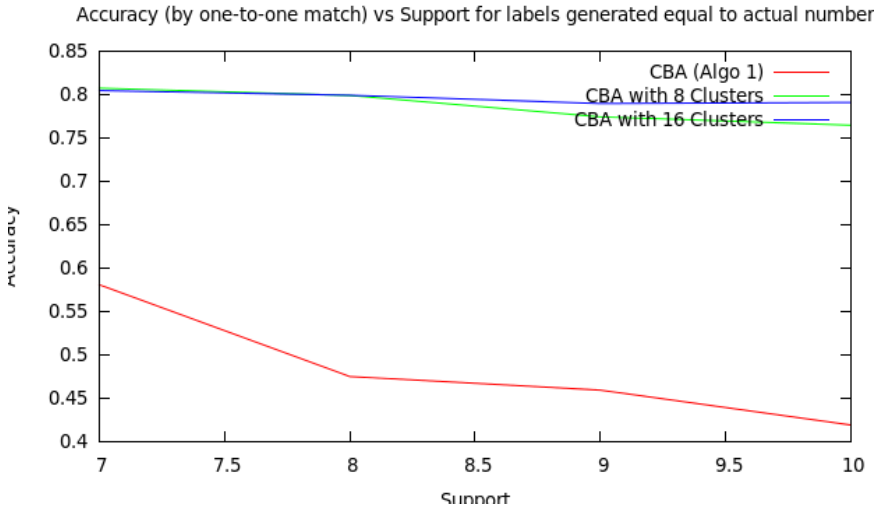
Following graphs show the same 4 accuracy measures as used for Algorithm 1, with fixed confidence value = 60%. It can be noticed that for all 3 cases (1 cluster, 8 clusters and 16 clusters), accuracy falls as the support goes on increasing. This is because the number of rules generated by apriori program is reduced and not all test transactions are correctly covered as a result.

An important thing to note is as the number of clusters increase, there is a significant gain in accuracy for the same support and confidence. Moreover, accuracy is less prone to decrease with increasing support in case of more clusters. This behavior can be explained by the fact that clustering groups similar transactions together and thus only relevant rules to the test transaction are generated and applied, giving a better classification quality.





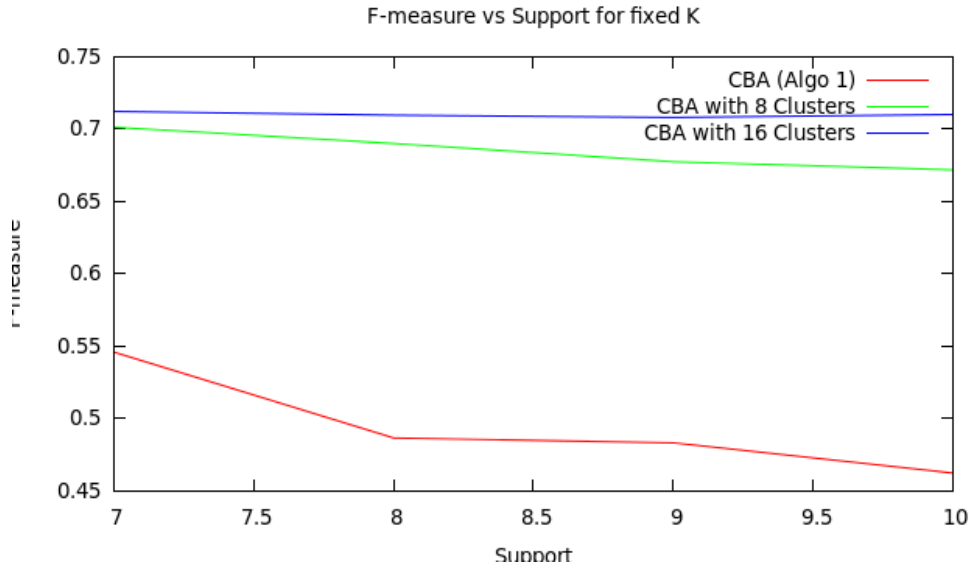


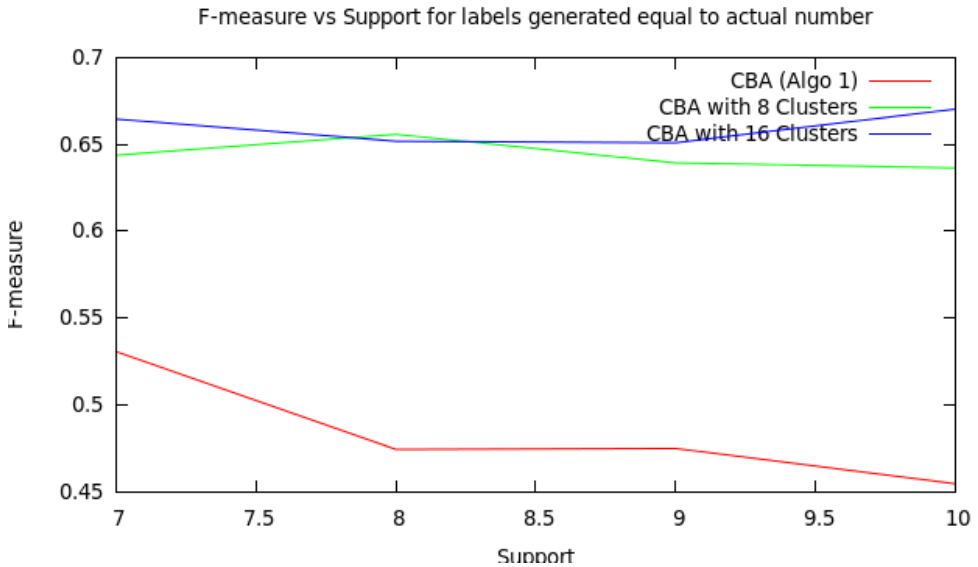


F-measure

We computed F-measure for the 3 cases. Following plots show the variation of F-measure over varying support and fixed confidence = 60%.

A similar trend as accuracy is observed in F-measure, i.e. F-measure tends to fall with increasing support, while it increases with increase in number of clusters. Same reasoning in accuracy applies to explain this behavior.

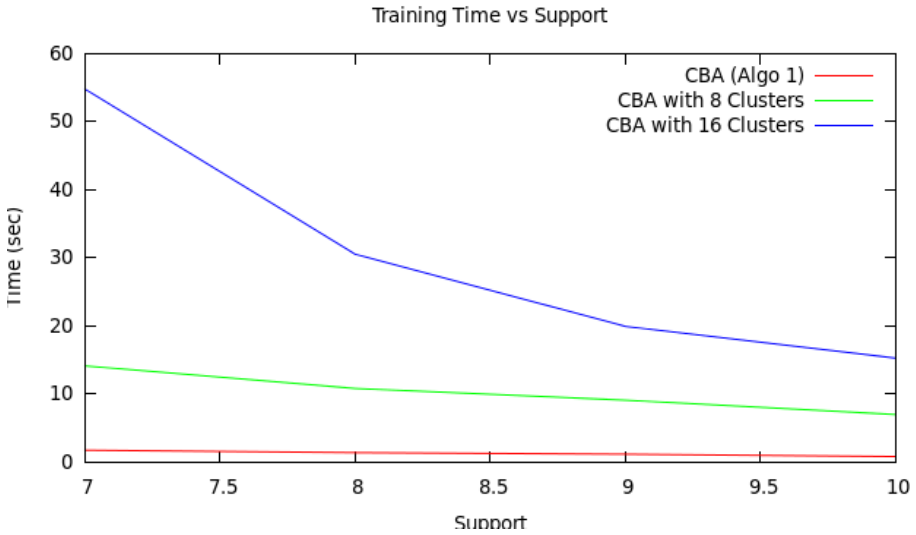




Training Time

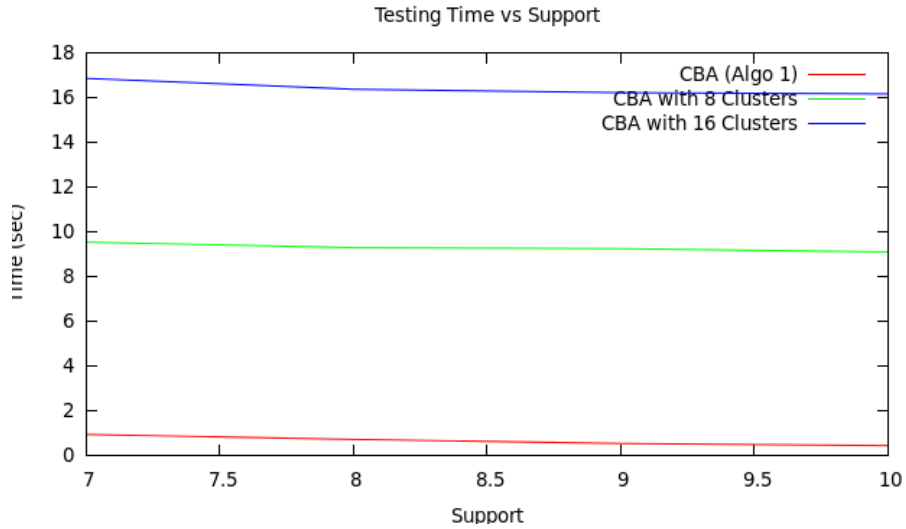
Training time is the time required to generate and process association rules. As support increases, the number of rules generated is reduced as reasoned before. And therefore, the training time drops.

Also, training time is more for more number of clusters. This is because as the number of clusters increase, the total number of rules (= sum of number of rules in each cluster) increases. This adds to the processing time. Again, there is a trade-off between speed and accuracy.



Testing time

The time to classify all transactions in the test set is testing time. The time to classify is more for more number of clusters due to more rules to scan through for applying them on the test instance. The gradual decrease in testing time with increased support is less than that in training time since the complexity of computations is same unlike in training phase with the additional time of identifying cluster for test instances which is negligible.



Tabular Results

The actual results have been provided in **Results.xlsx (4 sheets)** for reference.

Individual Contributions

Vaibhav implemented the basic algorithm for both algorithms. Akshay worked on clustering and acquiring results for both cases, implemented performance metrics as well as analyzed the performance for both algorithms.

REFERENCES:

[1] http://www.borgelt.net//apriori.html