CSC 869

Data Mining

Mini-Project 3

(Decision Tree Classifiers and Clustering)

Project Report

Submitted By:

Anshul Vyas

Spring 2015

SAN FRANCISCO STATE UNIVERSITY

SAN FRANCISCO, CA

Contents

PART I:Comparison of Decision Tree classifier and Naïve Bayesian classifier:

Using the adult dataset (http://archive.ics.uci.edu/ml/datasets/Adult) also known as Census Income dataset, we implement the C4.5 classifier to classify the dataset into <50K salary and >=50K salary.

Previous work on the dataset, while implementing Naïve Bayes classifier, yielded the accuracy of 76.07% while using a 10-fold Cross-Validation strategy.

Similar analysis is now to be done while building a decision-tree classifier using an existing implementation of C4.5 classifier. The options given were: J48 classifier available in Weka, and scikitlearn.tree.decisionTreeClassifier.

I made use of the J48 classifier provided in Weka. After applying 10-fold Cross validation strategy on the dataset, the results were as follows:

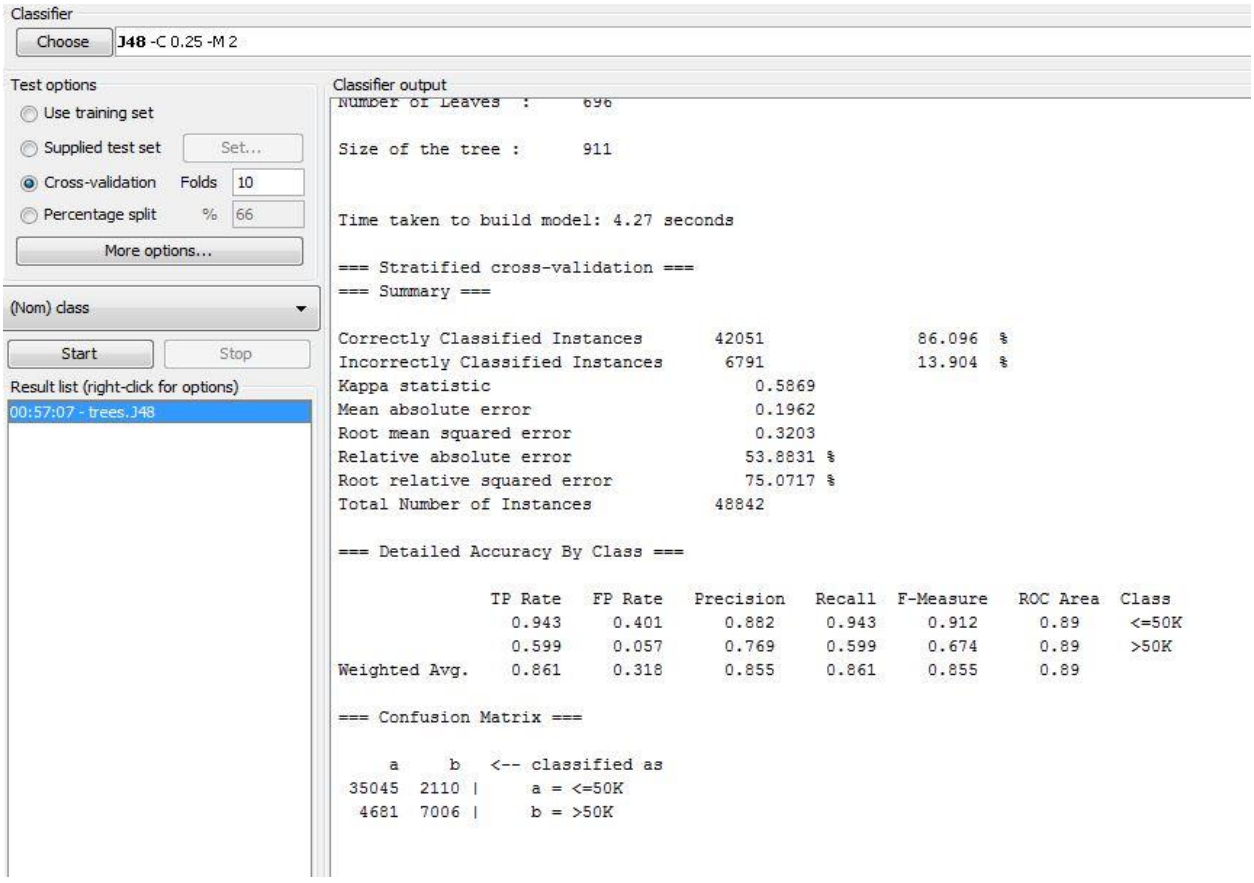

Figure 1 Evaluation of adult dataset using C4.5 classifier and 10-fold CV

The average accuracy for the classifier came out to be 86.03%

My implementation of Naïve Bayesian classifier with 10-fold Cross validation gave the following output:

```
no of True Positives: 12435
no of False Positives: 3846
Total: 16281
76.3773723973
48842
4884
{0: [0, 4884], 1: [4884, 9768], 2: [9768, 14652], 3: [14652, 19536], 4: [19536,
24420], 5: [24420, 29304], 6: [29304, 34188], 7: [34188, 39072], 8: [39072, 4395
6], 9: [43956, 48840]}
no of True Positives: 3699
no of False Positives: 1185
Total: 4884
75.7371007371
no of True Positives: 3747
no of False Positives: 1137
Total: 4884
76.7199017199
no of True Positives: 3712
no of False Positives: 1172
Total: 4884
76.0032760033
no of True Positives: 3715
no of False Positives: 1169
Total: 4884
76.0647010647
no of True Positives: 3694
no of False Positives: 1190
Total: 4884
75.6347256347
no of True Positives: 3700
no of False Positives: 1184
Total: 4884
75.7575757576
no of True Positives: 3682
no of False Positives: 1202
Total: 4884
75.389025389
no of True Positives: 3746
no of False Positives: 1138
Total: 4884
76.6994266994
no of True Positives: 3738
no of False Positives: 1146
Total: 4884
76.5356265356
no of True Positives: 3721
no of False Positives: 1163
Total: 4884
76.1875511876
mean accuracy: 76.0728910729 %
```

Figure 2 Evaluation of adult dataset with Naive Bayesian Classifier and with 10-fold Cross Validation

Comparing both classifiers:

When we compare my implementation of Naïve Bayesian classifier with the C4.5 classifier given in Weka, the mean accuracy of weka comes out to be significantly better than my classifier. Even though the input dataset for my classifier was all categorical, the correctly classified instances were less than that in the C4.5.

I believe my classifier was algorithmically not very optimized and that is the reason it took more time as well as classified less accurately.

PART II: Clustering information using Iris dataset:

For this, we used an existing dataset provided by Weka itself. The dataset consists of 150 instances which are divided into 3 categories: iris-setosa, iris-versicolor, iris-virginica. There are no missing values in the dataset. All the instances are divided into these categories equally, i.e, every class contains 50 instances from the whole dataset.

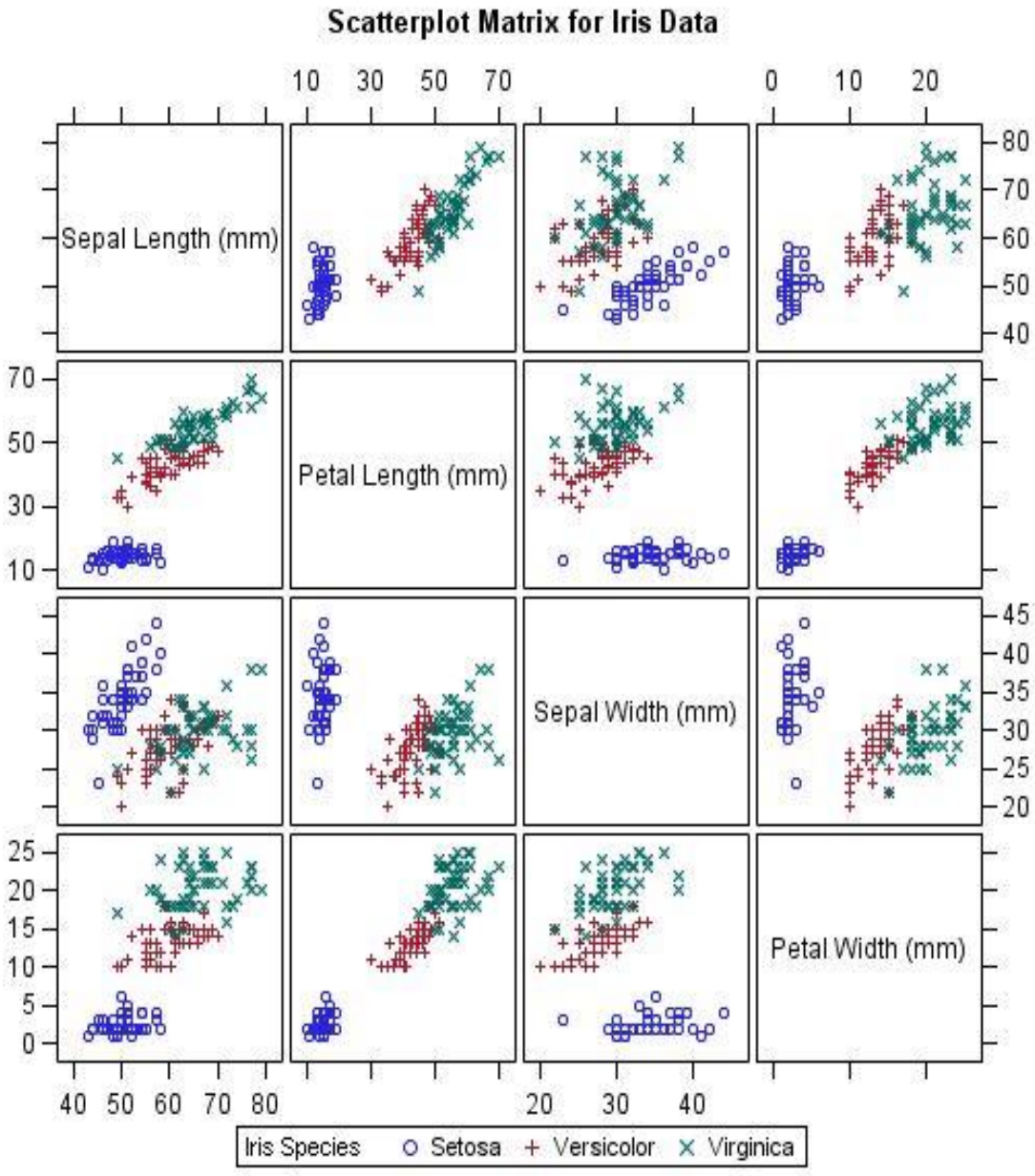

Figure 3 Iris Dataset

The graphical representation of the Iris dataset is given above, now we move on to the clustering algorithms, ie, SimpleKMeans, XMeans and DBSCAN.

1. Applying SimpleKMeans Algorithm to the dataset:

(i.) While applying simpleKmeans algorithm, the number of clusters I kept in the settings were 3. So the results of the evaluation were as follows:

```
=== Model and evaluation on training set ===

Clustered Instances

0       61  ( 41%)
1       50  ( 33%)
2       39  ( 26%)


Class attribute: class
Classes to Clusters:

  0  1  2  <-- assigned to cluster
  0 50  0 | Iris-setosa
 47  0  3 | Iris-versicolor
 14  0 36 | Iris-virginica

Cluster 0 <-- Iris-versicolor
Cluster 1 <-- Iris-setosa
Cluster 2 <-- Iris-virginica

Incorrectly clustered instances :       17.0     11.3333 %
```

Figure 4 Evaluation results of SimpleKMeans

If we see the clusters visually:

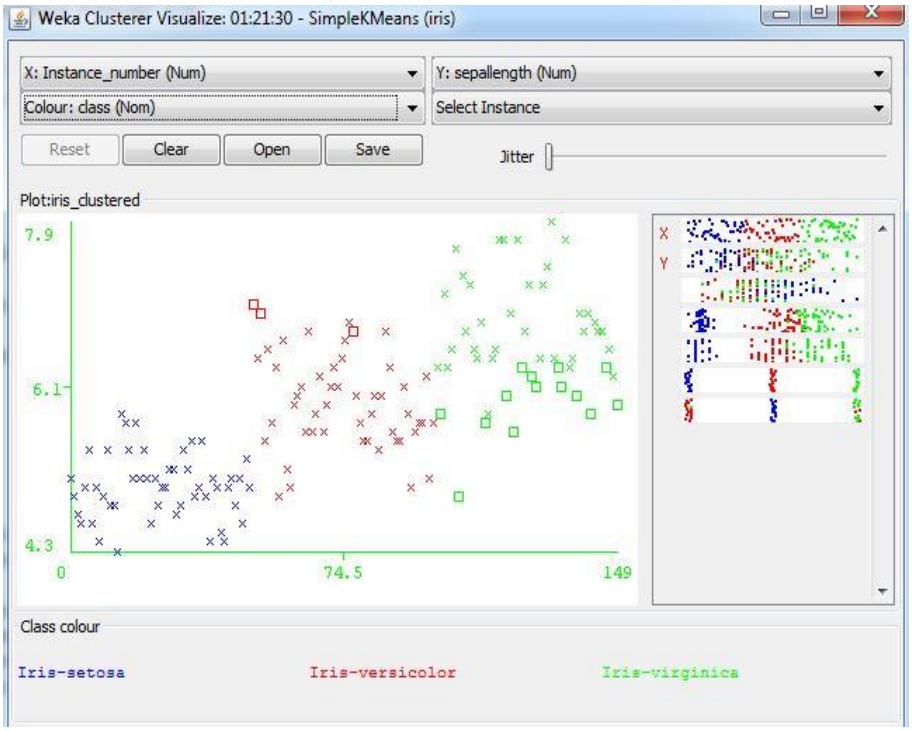

Figure 5 Visulization of clusters in SimpleKMeans

(ii.) External evaluation Measures:

(a.) Purity:

Classes to Clusters:

Cluster 0	Cluster 1	Cluster 2	<- Assigned to cluster
0	50	0	Iris – setosa
47	0	3	Iris – versicolor
14	0	36	Iris - virginica

Purity = 1/150 * (47 + 50 + 36)
= 0.887

Purity = 0.887

(b.) Normalized Mutual Information:

Here to calculate NMI, I made use of the normalized_mutual_info_score() method of python's sklearn kit.

```
from readARFF import *
import numpy as np
from sklearn.metrics.cluster import normalized_mutual_info_score

def calcNMI():

    dataset = readARFF();

    subSet = dataset[['class', 'cluster']]
    print subSet

    NMI = normalized_mutual_info_score(subSet['class'], subSet['cluster'])
    print NMI

calcNMI()
```

NMI = 0.741

(c.) The Rand Index

For calculating the Rand index we first need to find the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

All pairs of documents = N (N-1)/2 = 150 (150 – 1)/2
= 11175

Total Positives = TP + FP = ${}^{61}C_2 + {}^{50}C_2 + {}^{39}C_2 = 3796$

TP = ${}^{47}C_2 + {}^{14}C_2 + {}^{50}C_2 + {}^{3}C_2 + {}^{39}C_2 = 3030$

FP = (TP + FP) – TP = 3796 – 3030 = 766

Total Negatives = N (N – 1)/2 – total positives = 11175 – 3796
= 7379

FN = (47 * 3) + (14 * 36) = 645

TN = Total Negatives – FN = 7379 – 645 = 6734

The final confusion matrix is:

	Same Cluster	Different Clusters
Same Class	**TP = 3030**	**FN = 645**
Different Classes	**FP = 766**	**TN = 6734**

Now the Rand index,

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

RI = (3030 + 6734) / (3030 + 766 + 645 + 6734)
= 0.874

Rand Index = 0.874

(d.)The F1 Measure ($\beta = 1$)

For F1 measure, we will first calculate the Precision and Recall:

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

P = 3030/ (3030 + 766)

= 0.799

R = 3030 / (3030 + 645)

= 0.824

F1 = 2PR/(P + R)

= 2 (0.799 * 0.824) / (0.799 + 0.824)

= 0.811

Precision = 0.799

Recall = 0.824

F1 Measure = 0.811

(iii.) Internal evaluation Measures:

For calculating the diameters of the clusters and also the average link and complete link, I wrote a python script which loops over all the points and calculate the Euclidean distance between each one of them, then applying the Diameter formula:

$$D_m = \sqrt{\frac{\Sigma_{i=1}^{N} \Sigma_{i=1}^{N} (t_{ip} - t_{iq})^2}{N(N-1)}}$$

Dividing the python dataframe according to cluster values, I was able to calculate both the average link and the complete link, the results are recorded in the table below:

COMPLETE LINK METHOD:

```
"""@return complete link between two clusters"""
def completeLink(list1, list2, dimens=4):
    distLarge = 0

    for x in range(0, len(list1)):
        for y in range(0, len(list2)):
            distNow = calcEuclideanDist(list1[x], list2[y], 4)
            if distNow > distLarge:
                distLarge = distNow

    return distLarge
```

Summary of evaluations of SimpleKMeans:

For SimpleKMeans algorithm,

Purity = 0.887
NMI = 0.741
Rand Index = 0.874
F1 measure $(\beta = 1)$ = 0.811

Diameter of cluster 0 = 1.13070411836
Diameter of cluster 1 = 0.78870599266
Diameter of cluster 2 = 1.16591261539

Average Link Cluster 0 to 1 = 3.43497254041
Average Link Cluster 1 to 2 = 5.03866720344
Average Link Cluster 0 to 2 = 1.94200758131

Complete Link Cluster 0 to 1 = 5.08428952755
Complete Link Cluster 1 to 2 = 7.08519583357
Complete Link Cluster 0 to 2 = 4.83942145303

2. Applying XMeans Algorithm to the dataset:

While applying XMeans algorithm, the minimum number of clusters I kept were 3. So the results of the evaluation were as follows:

```
=== Model and evaluation on training set ===

Clustered Instances

0       52  ( 35%)
1       48  ( 32%)
2       50  ( 33%)


Class attribute: class
Classes to Clusters:

  0  1  2  <-- assigned to cluster
  0  0 50 | Iris-setosa
 10 40  0 | Iris-versicolor
 42  8  0 | Iris-virginica

Cluster 0 <-- Iris-virginica
Cluster 1 <-- Iris-versicolor
Cluster 2 <-- Iris-setosa

Incorrectly clustered instances :       18.0     12      %
```

Figure 6 Evaluation results of XMeans

If we see the clusters visually:

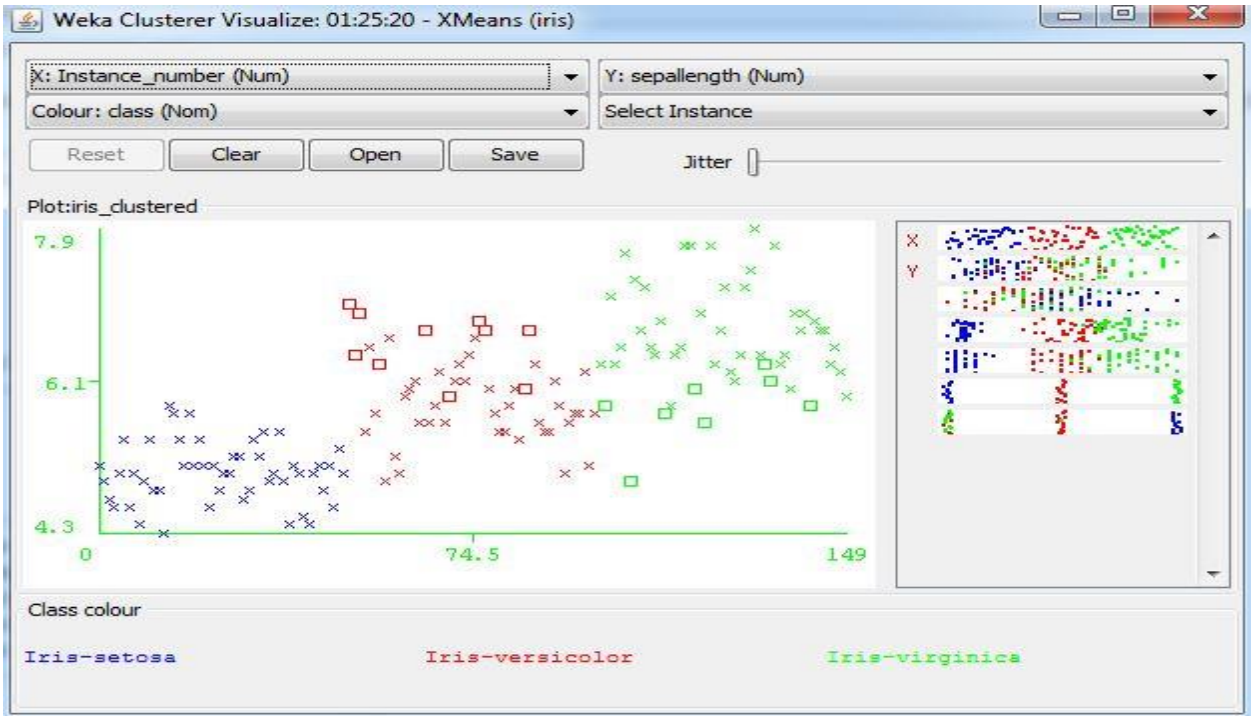

Figure 7 Visualization of clusters in XMeans

(iv.) External evaluation Measures:

(a.) Purity:

Classes to Clusters:

Cluster 0	Cluster 1	Cluster 2	<- Assigned to cluster
0	0	50	Iris – setosa
10	40	0	Iris – versicolor
42	8	0	Iris - virginica

Purity = 1/150 * (42 + 40 + 50)
= 0.88

Purity = 0.88

(b.) Normalized Mutual Information:

Using the same python script as above:

NMI = 0.714

(c.) The Rand Index

For calculating the Rand index we first need to find the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

All pairs of documents = N (N-1)/2 = 150 (150 – 1)/2
= 11175

$TP + FP = {}^{52}C_2 + {}^{48}C_2 + {}^{50}C_2 = 3679$

$TP = {}^{10}C_2 + {}^{42}C_2 + {}^{40}C_2 + {}^{8}C_2 + {}^{50}C_2 = 2939$

FP = (TP + FP) – TP = 3679 – 2939 = 740

Total Negatives = N (N – 1)/2 – total positives = 11175 – 3679
= 7496

FN = (40 * 10) + (42 * 8) = 736

TN = Total Negatives – FN = 7496 – 736 = 6760

The final confusion matrix is:

	Same Cluster	Different Clusters
Same Class	**TP = 2939**	**FN = 736**
Different Classes	**FP = 740**	**TN = 6760**

Now the Rand index,

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

RI = (2939 + 6760) / (2939 + 740 + 736 + 6760)
= 0.868

Rand Index = 0.868

(d.)The F1 Measure

For F1 measure, we will first calculate the Precision and Recall:

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN} \qquad F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

P = 2939/ (2939 + 740)

= 0.799

R = 2939 / (2939 + 736)

= 0.799

F1 = 2PR/(P + R)

= 2 (0.799 * 0.799) / (0.799 + 0.799)

= 0.799

Precision = 0.799

Recall = 0.799

F1 Measure = 0.799

(v.) Internal evaluation Measures:

I ran the same scripts as shown above in the SimpleKMeans algorithm by replacing the ARFF file, and the results for XMeans are recorded in the table below:

AVERAGE LINK METHOD:

```
"""@return average link betwee two clusters"""
def averageLink(list1, list2, dimens=4):
    distSum = 0
    N = 0

    for x in range(0, len(list1)):
        for y in range(0, len(list2)):
            distNow = calcEuclideanDist(list1[x], list2[y], 4)
            distSum = distSum + distNow
            N = N + 1

    averageLink = distSum/N
    return averageLink
```

Summary of evaluations of XMeans,

For XMeans algorithm,

Purity = 0.88
NMI = 0.714
Rand Index = 0.868
F1 measure $(\beta = 1)$ = 0.799

Diameter of cluster 0 = 1.28439408865
Diameter of cluster 1 = 1.10446386339
Diameter of cluster 2 = 0.78870599266

Average Link Cluster 0 to 1 = 1.84988198308
Average Link Cluster 1 to 2 = 3.31231015508
Average Link Cluster 0 to 2 = 4.75097035491

Complete Link Cluster 0 to 1 = 4.83942145303
Complete Link Cluster 1 to 2 = 5.08428952755
Complete Link Cluster 0 to 2 = 7.08519583357

3. A density-based clustering algorithm: DBSCAN (default values)

While applying the DBSCAN algorithm, using the default values of *esp = 0.9* and *minPts = 6*, the results of the evaluation were,

```
=== Model and evaluation on training set ===

Clustered Instances

0      150 (100%)


Class attribute: class
Classes to Clusters:

  0  <-- assigned to cluster
 50 | Iris-setosa
 50 | Iris-versicolor
 50 | Iris-virginica

Cluster 0 <-- Iris-setosa

Incorrectly clustered instances :       100.0     66.6667 %
```

Figure 8 Evaluation results of the DBSCAN algorithm for the IRIS dataset

If we see cluster visually:

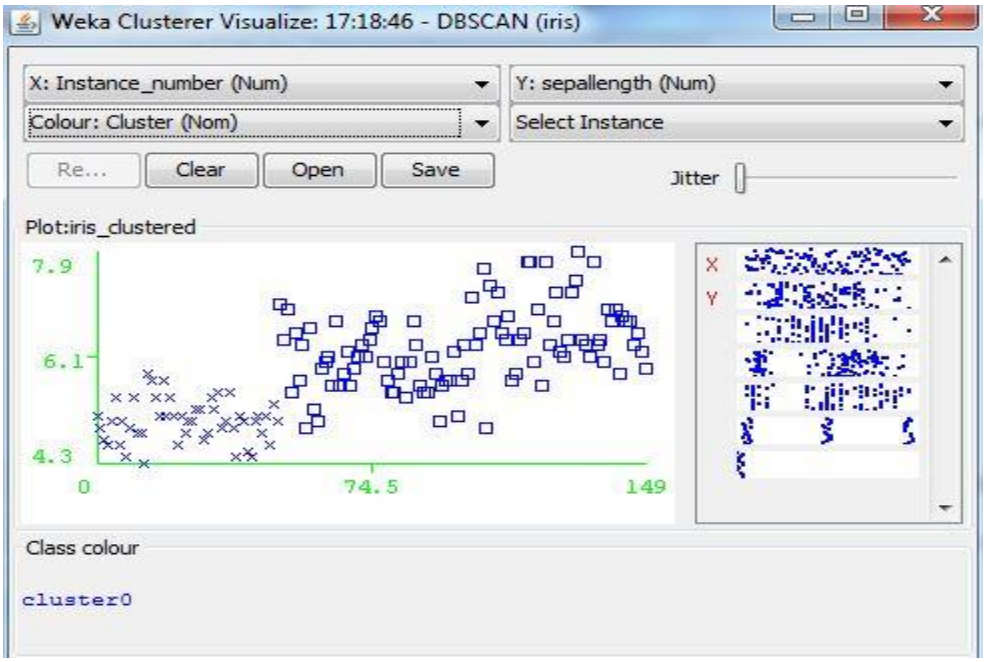

Figure 9 Visualization of DBSCAN algorithm

(vi.) External evaluation Measures:

(a.) Purity:

Classes to Cluster:

Cluster 0	<- Assigned to cluster
50	Iris – setosa
50	Iris – versicolor (incorrect)
50	Iris – virginica (incorrect)

Purity = 1/150 * (50)
= 0.33

Purity = 0.33

(b.) The Rand Index

For calculating the Rand index we first need to find the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

All pairs of documents = N (N-1)/2 = 150 (150 – 1)/2
= 11175

$TP = {}^{50}C_2 + {}^{50}C_2 + {}^{50}C_2 = 3675$

FP = N (N – 1)/2 – TP = 11175 – 3675 = 7500

Total Negatives = 0

The final confusion matrix is:

	Same Cluster	Different Clusters
Same Class	**TP = 3675**	**FN = 0**
Different Classes	**FP = 7500**	**TN = 0**

Now the Rand index,

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

RI = (3675 + 0) / (3675 + 7500 + 0 + 0)
= 0.33

Rand Index = 0.33

(c.) The F1 Measure

For F1 measure, we will first calculate the Precision and Recall:

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN} \qquad F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

P = 3675/ (3675 + 7500)

= 0.33

R = 3675 / (3675 + 0)

= 1

F1 = 2PR/(P + R)

= 2 (0.33 * 1) / (0.33 + 1)

= 0.799

Precision = 0.33

Recall = 1

F1 Measure = 0.48

(vii.) Internal evaluation Measures:

I ran the same script as the SimpleKMeans algorithm by replacing the ARFF file, as there was only 1 cluster, there was no need for splitting the dataframe. The results for DBSCAN are recorded in the table below:

DIAMETER METHOD:

```
"""@return the diameter of a particular cluster"""
def getDiameter(dataFrame):
    size = len(dataFrame)
    print size

    myList = dataFrame.values.tolist()
    #print myList
    diam = 0
    N = size
    for i in range(0,N):
        for j in range(0,N):
            if i is not j:
                diam = diam + (calcEuclideanDist(myList[i],myList[j])**2)

    return math.sqrt(diam/(N*(N-1)))
```

Summary of evaluations of DBSCAN,

For DBSCAN algorithm, (default value – *esp = 0.9 minpts = 6*)

Purity = 0.33
NMI = 0
Rand Index = 0.33
F1 measure *(β = 1)* = 0.48

Diameter of cluster = 3.02300885714

4. DBSCAN (custom values - *esp = 0.285 minpts = 12***):**

If, however, we play with the values of *esp = 0.285 minpts = 12*, we get 2 clusters instead of 1 and all the evaluation measures increase:

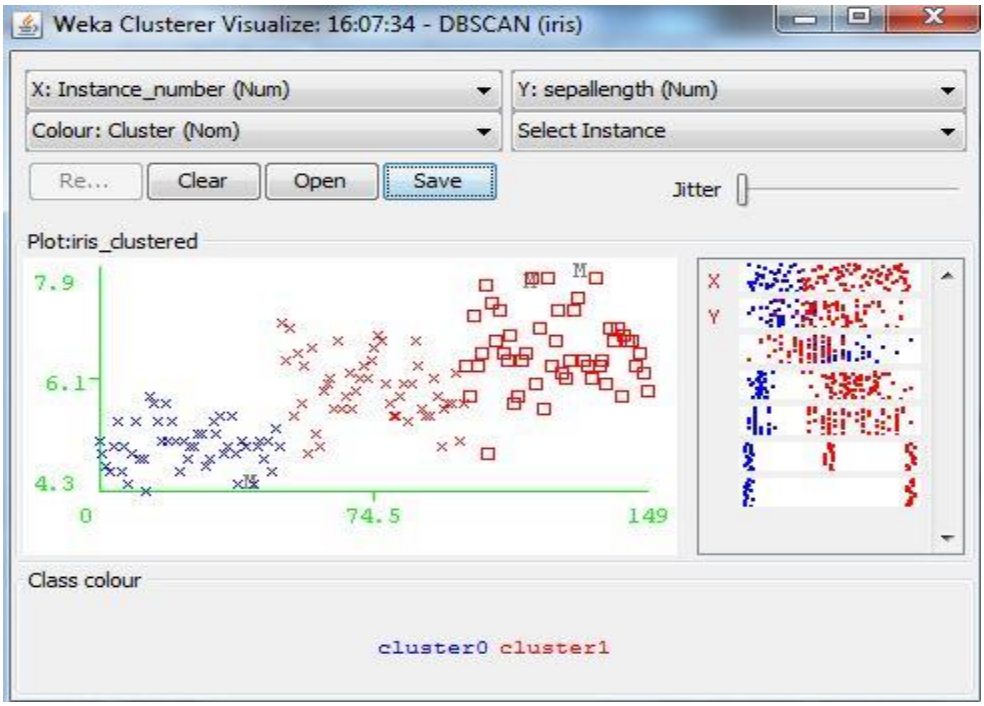

Figure 10 DBSCAN (esp = 0.285, minPts = 6)

The evaluation results were:

```
=== Model and evaluation on training set ===

Clustered Instances

0       50 ( 34%)
1       98 ( 66%)

Unclustered instances : 2

Class attribute: class
Classes to Clusters:

  0  1  <-- assigned to cluster
 50  0 | Iris-setosa
  0 50 | Iris-versicolor
  0 48 | Iris-virginica

Cluster 0 <-- Iris-setosa
Cluster 1 <-- Iris-versicolor

Incorrectly clustered instances :       48.0     32      %
```

Figure 11 Evaluation results of DBSCAN custom values

So for the custom values the internal and external evaluations measures are as follows:

For DBSCAN algorithm, (custom values value – *esp = 0.285 minpts = 12*)

Purity = 0.66
NMI = 0.735543502353
Rand Index = 0.785
F1 measure *(β = 1)* = 0.742

Diameter of cluster 0 = 0.78870599266
Diameter of cluster 1 = 1.61974522209

Average Link Cluster 0 to 1 = 4.018259073169099
Complete Link Cluster 0 to 1 = 7.085195833567341

OBSERVATIONS:

We have run different clustering algorithms on the same dataset (IRIS). The following table compares all the algorithms based on their internal and external evaluations:

SimpleKMeans	XMeans	DBSCAN (default values *esp = 0.9 minpts = 6*)	DBSCAN (custom values *esp = 0.285 minpts = 12*)
External evaluations: Purity = 0.887 NMI = 0.741 Rand Index = 0.874 F1 measure *(β = 1)* = 0.811	**External evaluations:** Purity = 0.88 NMI = 0.714 Rand Index = 0.868 F1 measure *(β = 1)* = 0.799	**External evaluations:** Purity = 0.33 NMI = 0 Rand Index = 0.33 F1 measure *(β = 1)* = 0.48	**External evaluations:** Purity = 0.66 NMI = 0.735543502353 Rand Index = 0.785 F1 measure *(β = 1)* = 0.742
Internal evaluations: Diameter cluster 0 = 1.13070411836 Diameter cluster 1 = 0.78870599266 Diameter cluster 2 = 1.16591261539 Average Link Cluster 0 to 1 = 3.43497254041 Average Link Cluster 1 to 2 = 5.03866720344 Average Link Cluster 0 to 2 = 1.94200758131 Complete Link Cluster 0 to 1 = 5.08428952755 Complete Link Cluster 1 to 2 = 7.08519583357 Complete Link Cluster 0 to 2 = 4.83942145303	**Internal evaluations:** Diameter of cluster 0 = 1.28439408865 Diameter of cluster 1 = 1.10446386339 Diameter of cluster 2 = 0.78870599266 Average Link Cluster 0 to 1 = 1.84988198308 Average Link Cluster 1 to 2 = 3.31231015508 Average Link Cluster 0 to 2 = 4.75097035491 Complete Link Cluster 0 to 1 = 4.83942145303 Complete Link Cluster 1 to 2 = 5.08428952755 Complete Link Cluster 0 to 2 = 7.08519583357	**Internal evaluations:** Diameter of cluster = 3.02300885714	**Internal evaluations:** Diameter of cluster 0 = 0.78870599266 Diameter of cluster 1 = 1.61974522209 Average Link Cluster 0 to 1 = 4.018259073169099 Complete Link Cluster 0 to 1 = 7.085195833567341

The graphical representation of the external evaluations of the four algorithms is:

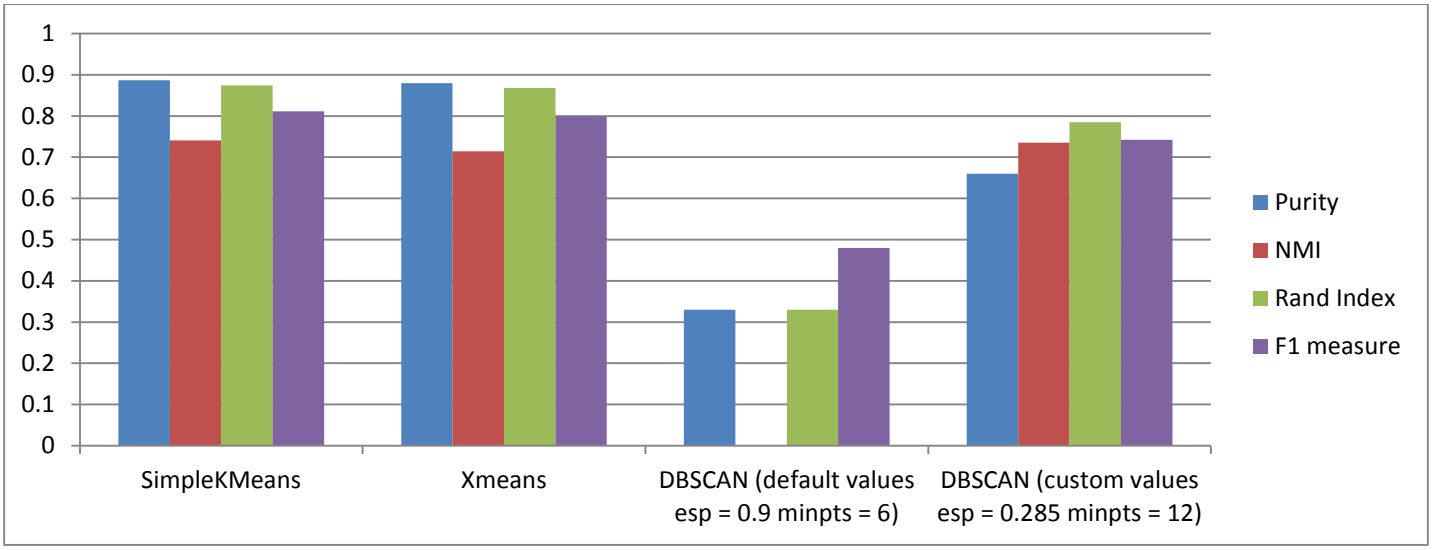

Figure 12 Comparison of External evaluation measures of different algorithms

As we see from the bar plots, the values of external evaluation factors in SimpleKMeans and XMeans is relatively better than the DBSCAN algorithm. However, when we take custom values of DBSCAN (*esp = 0.285 minpts = 12)*, the values are much better than the default *esp* and *minPts* values.

Moreover, the advantage of DBSCAN over SimpleKMeans and XMeans is that there is no need to define an external factor k, explicitly, which is a disadvantage of both SimpleKMeans and XMeans.