YILDIZ Technical University
Computer Engineering Department

Introduction to Data Mining — Semester Project

Student Name: Berk Sudan
Student Number: 16011601

Due Date: 18.12.2018

Signature:

[illegible]

1. Work On WEKA

1.a. Data Analysis

Data Content

The name of dataset is “Credit Approval”. The area of this dataset is financial. The database consists of instances from different users’ credit card applications in a bank. The motivation of this data is to check the approval of the user to get the credit card or not. Since bank employees are not capable of checking all users’ credit card application manually, this data mining program will help bank to decide if they should give credit card to a customer or not.

In dataset information, It is stated that the collection of dataset is unknown in order to protect confidentiality. The data was probably collected from the database of a bank. Characteristics of data set is multivariate because of the fact that it has nominal and numerical attributes

Attributes

All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. Hence, we can never know what do the attributes mean. In spite of that, we don’t need attribute name for classification or clustering.

There are 16 attributes of the provided data. The types of the attributes are shown below:

1. First Attribute:
    - Categorical data, 2 categories.
    - Values are: {a,b}
2. Second Attribute:
    - Continues real (floating number) data.
3. Third Attribute:
    - Continues real (floating number) data.
4. Fourth Attribute:
    - Categorical data, 3 categories.
    - Values are: {l,u,y}

5. Fifth Attribute:
    - Categorical data, 3 categories.
    - Values are: {g,gg,p}
6. Sixth Attribute:
    - Categorical data, 14 categories.
    - Values are: {aa,c,cc,d,e,ff,i,j,k,m,q,r,w,x}
7. Seventh Attribute:
    - Categorical data, 9 categories.
    - Values are: {bb,dd,ff,h,j,n,o,v,z}
8. Eighth Attribute:
    - Continues real (floating number) data.
9. Ninth Attribute:
    - Categorical data, 2 categories.
    - Values are: {t,f}
10. Tenth Attribute:
    - Categorical data, 2 categories.
    - Values are: {t,f}
11. Eleventh Attribute:
    - Continues real (floating number) data.
12. Twelfth Attribute:
    - Categorical data, 2 categories.
    - Values are: {t,f}
13. Thirteenth Attribute:
    - Categorical data, 3 categories.
    - Values are: {g, p, s}
14. Fourteenth Attribute:
    - Continues real (floating number) data.

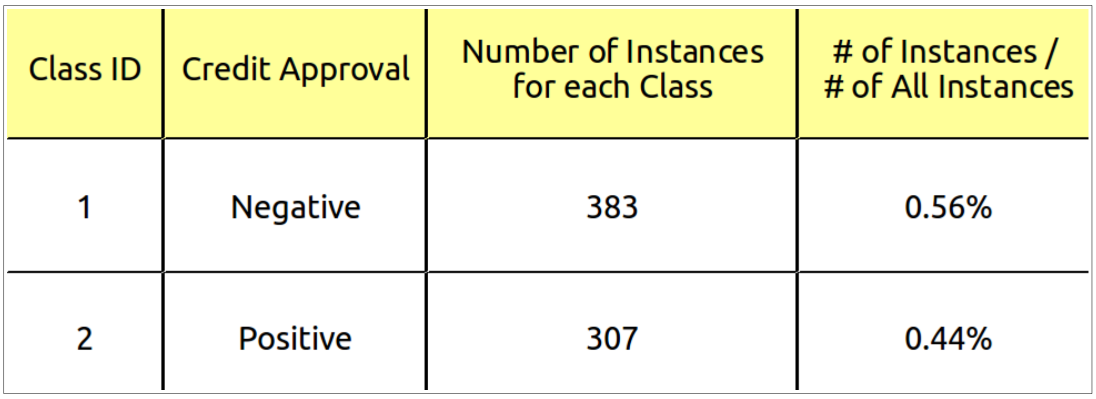
15. Fifteenth Attribute:
    - Continues real (floating number) data.
16. Class (Credit_Approval)
    - Categorical data, 2 categories.
    - Values are: {+, -}

Attribute 16 is **output attribute** (a.k.a. class). Attributes from 1 to 15 are **input attributes**.

Class Attribute & Class Labels

Each class label represents the status of credit card application approval. There are **2 class** labels in total**.** If class label is "+", then the customer can get credit card. Otherwise, the customer can't get credit card. Dataset has **690 instances**.

These class labels, number of instances per each class and percentage of number of instances of each class over all number of instances are shown below:



Class ID	Credit Approval	Number of Instances for each Class	# of Instances / # of All Instances
1	Negative	383	0.56%
2	Positive	307	0.44%

Sample standard deviation of the 4th column (which is # of instances / # of all instances) is approximately 7.788 and the mean is ~50.0. Hence, we can say that this dataset is **balanced**.

Missing Values

There are **67 missing values** in total and missing values are represented with “?” in the dataset. In each row, the number of extant values are much more greater than the number of missing values. So, we don’t need to delete any row/ instance.

In order to obtain missing values we will use different methods for numerical and categorical attributes:

- For numerical attributes, “Replace with its mean” method will be used.
- For categorical attributes, we will use the frequency of each category as the missing value’s possibility.

Calculation of the new values is explained below:

- <u>For attribute 1:</u> (Categorical)
  - 12 * 468 / 678 ---------→ |b| = 8, |a| :4
- <u>For attribute 2:</u> (Numerical)
  - Used the mean of the attribute which is 31.56817 .
- <u>For attribute 4:</u> (Categorical)
  - 6 * 519 / 684= “u” ---------→ |u| = 5, |y| = 1
- <u>For attribute 5:</u> (Categorical)
  - 6 * 519 / 684= “g” ---------→ |g| = 5, |p| = 1
- <u>For attribute 6:</u> (Categorical)
  - 9 * 137/ 681 = 1.8106 ---------→ |c| = 2
  - 9 * 78/ 681 = 1.0308 ---------→ |q| = 1
  - 9 * 64/ 681 = 0.8458 ---------→ |w| = 1
  - 9 * 59/ 681 = 0.7797 ---------→ |i| = 1
  - 9 * 54/ 681 = 0.7137 ---------→ |aa| = 1
  - 9 * 53/ 681 = 0.7004 ---------→ |ff| = 1
  - 9 * 51/ 681 = 0.6740 ---------→ |k| = 1
  - 9 * 41/ 681 = 0.5418 ---------→ |cc| = 1
  - 9 * 38/ 681 = 0.5022 ---------→ |x| = 0
  - 9 * 38/ 681 = 0.5022 ---------→ |m| = 0

- 9 * 30/ 681 = 0.3965 ---------→ |d| = 0
- 9 * 25/ 681 = 0.3304 ---------→ |e| = 0
- 9 * 10 / 681 = 0.1321 ---------→ |j| = 0
- 9 * 3 / 681 = 0.0396 ---------→ |r| = 0

◆ For attribute 7: (Categorical)

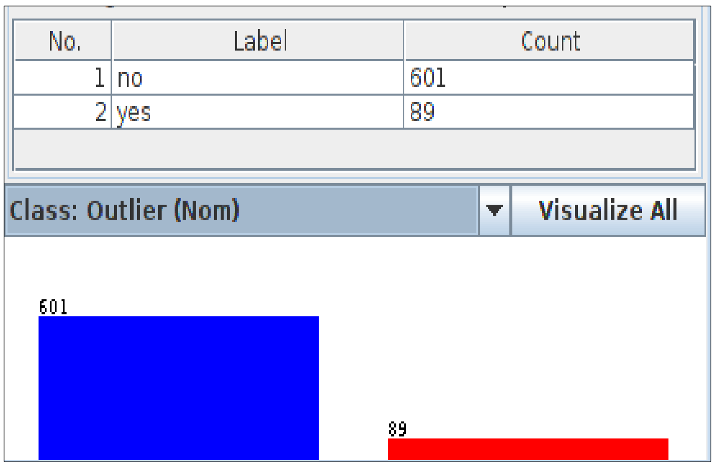
- 9 * 399 / 681 = 5.2731 ---------→ |v| = 5
- 9 * 138 / 681 = 1.8238 ---------→ |h| = 2
- 9 * 59 / 681 = 0.7797 ---------→ |bb| = 1
- 9 * 57 / 681 = 0.7533 ---------→ |ff| = 1
- 9 * 8 / 681 ---------→ |z| = 0
- 9 * 8 / 681 ---------→ |j| = 0
- 9 * 6 / 681 ---------→ |dd| = 0
- 9 * 4 / 681 ---------→ |n| = 0
- 9 * 2 / 681 ---------→ |o| = 0

◆ For attribute 14: (Numerical)

- Used the mean of the attribute which is 184.01477 .

Outlier Analysis

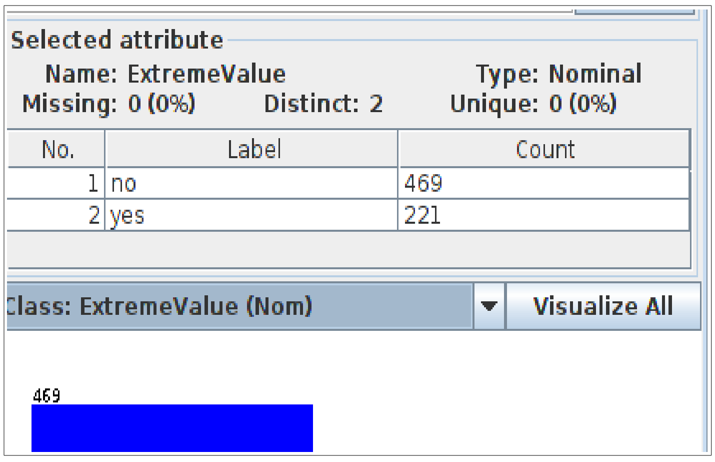
In order to detect outliers and extreme values, **Interquartile Range filter** (which is named as “weka.filters.unsupervised.attribute.InterquartileRange” in WEKA) is used. Afterward, we get 89 outliers and 221 extreme values which is shown below:



No.	Label	Count
1	no	601
2	yes	89

Class: Outlier (Nom)

Non-outlier & Outlier Instances



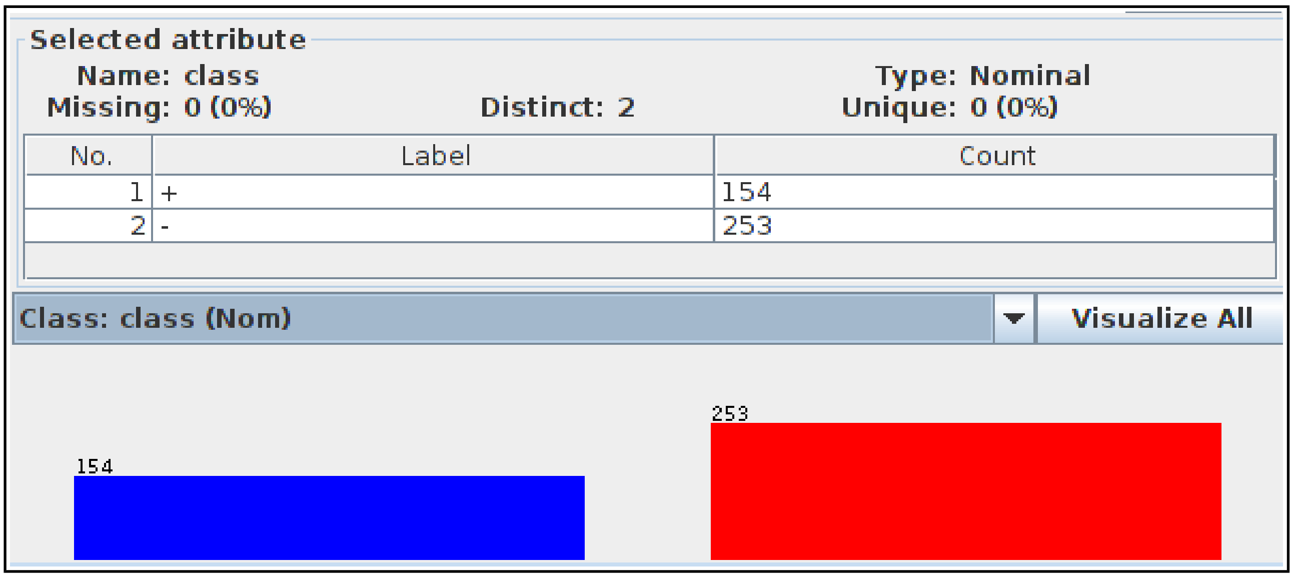
Selected attribute
Name: ExtremeValue Type: Nominal
Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

No.	Label	Count
1	no	469
2	yes	221

Class: ExtremeValue (Nom)

Non-extreme & Extreme Instances

After the removal of extreme values and outliers, we end up with **407 instances**, which is shown below:


Selected attribute
Name: class
Type: Nominal
Missing: 0 (0%)
Distinct: 2
Unique: 0 (0%)

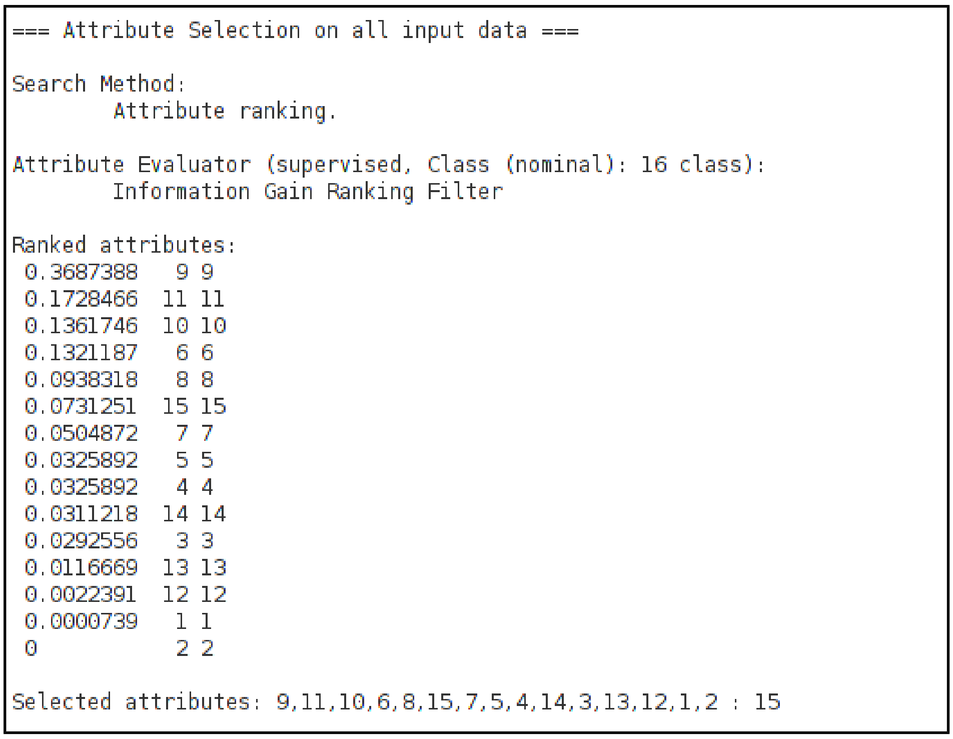
No.	Label	Count
1	+	154
2	-	253

Moreover, this reduced data is also considered to be as **balanced** because of the standard deviation of the frequency is ~3.016 . So, no need to do any subsampling operation.

Distinguishing Attributes

Information Gain Attribute Evaluation method is applied on the data and most distinguishing attribute to least distinguishing one are shown respectively: 9,11,10,6,8,15,7,5,4,14,3,13,12,1,2 .

The full result of the operation is shown below:


```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 16 class):
        Information Gain Ranking Filter

Ranked attributes:
 0.3687388    9 9
 0.1728466   11 11
 0.1361746   10 10
 0.1321187    6 6
 0.0938318    8 8
 0.0731251   15 15
 0.0504872    7 7
 0.0325892    5 5
 0.0325892    4 4
 0.0311218   14 14
 0.0292556    3 3
 0.0116669   13 13
 0.0022391   12 12
 0.0000739    1 1
 0            2 2

Selected attributes: 9,11,10,6,8,15,7,5,4,14,3,13,12,1,2 : 15
```

1.b. Classification

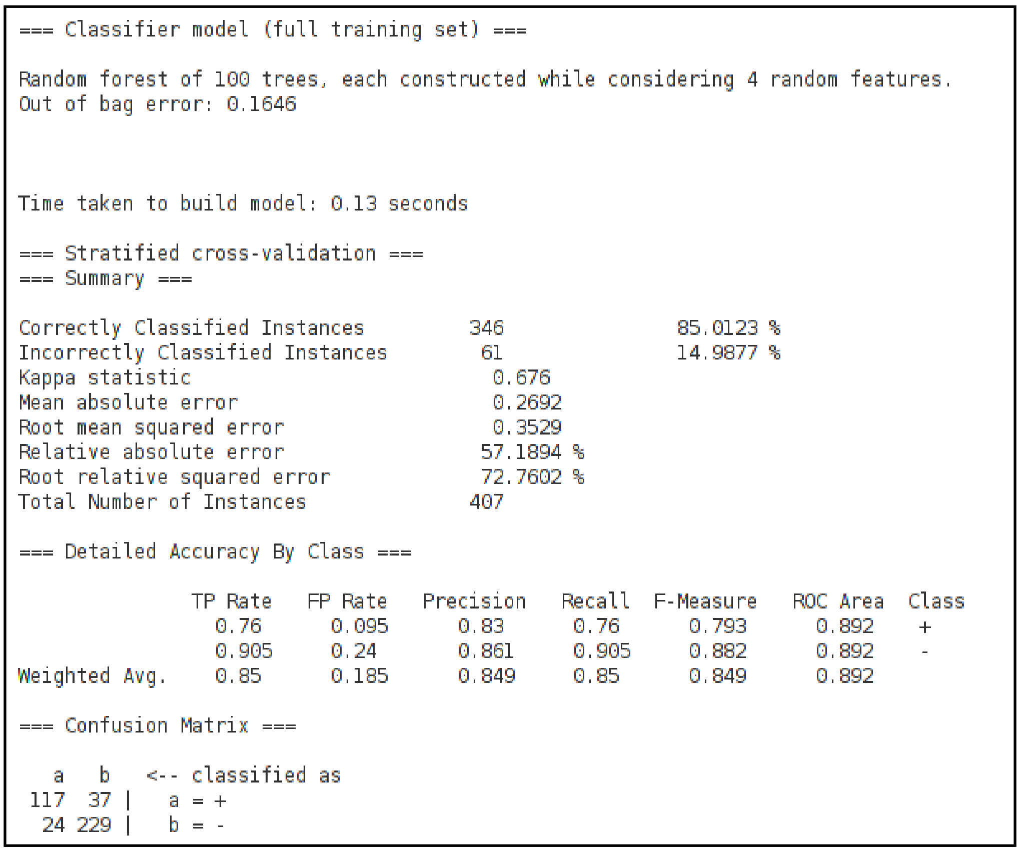
According to results of WEKA tool, the most successful algorithms are respectively:

1. Random Forest.
2. K Nearest Neighbors.
3. Random Comittee.

1. First Algorithm: Random Forest

Random forests, also known as random decision forests, are a popular ensemble method that can be used to build predictive models for both classification and regression problems. Ensemble methods use multiple learning models to gain better predictive results — in the case of a random forest, the model creates an entire forest of random uncorrelated decision trees to arrive at the best possible answer.

In WEKA tool, Random Forest Classification is present with name “weka.classifiers.trees.RandomForest”. In WEKA, k-fold cross validation method is used to determine training and test sets. Number of trees is **100** and seed is **1**. Different k values are selected for k-fold cross validation method. For **k = 2 (2-fold)** we had the best results with accuracy rate **85.01 %**. Result is shown below:



```
=== Classifier model (full training set) ===

Random forest of 100 trees, each constructed while considering 4 random features.
Out of bag error: 0.1646



Time taken to build model: 0.13 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         346               85.0123 %
Incorrectly Classified Instances        61               14.9877 %
Kappa statistic                          0.676
Mean absolute error                      0.2692
Root mean squared error                  0.3529
Relative absolute error                 57.1894 %
Root relative squared error             72.7602 %
Total Number of Instances              407

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.76      0.095      0.83      0.76      0.793      0.892    +
                 0.905     0.24       0.861     0.905     0.882      0.892    -
Weighted Avg.    0.85      0.185      0.849     0.85      0.849      0.892

=== Confusion Matrix ===

   a   b   <-- classified as
 117  37 |   a = +
  24 229 |   b = -
```

According to results, **346 instances** are **correctly classified** and **61 instances** are **incorrectly classified**.

For other k-fold values, accuracy rates are given below:

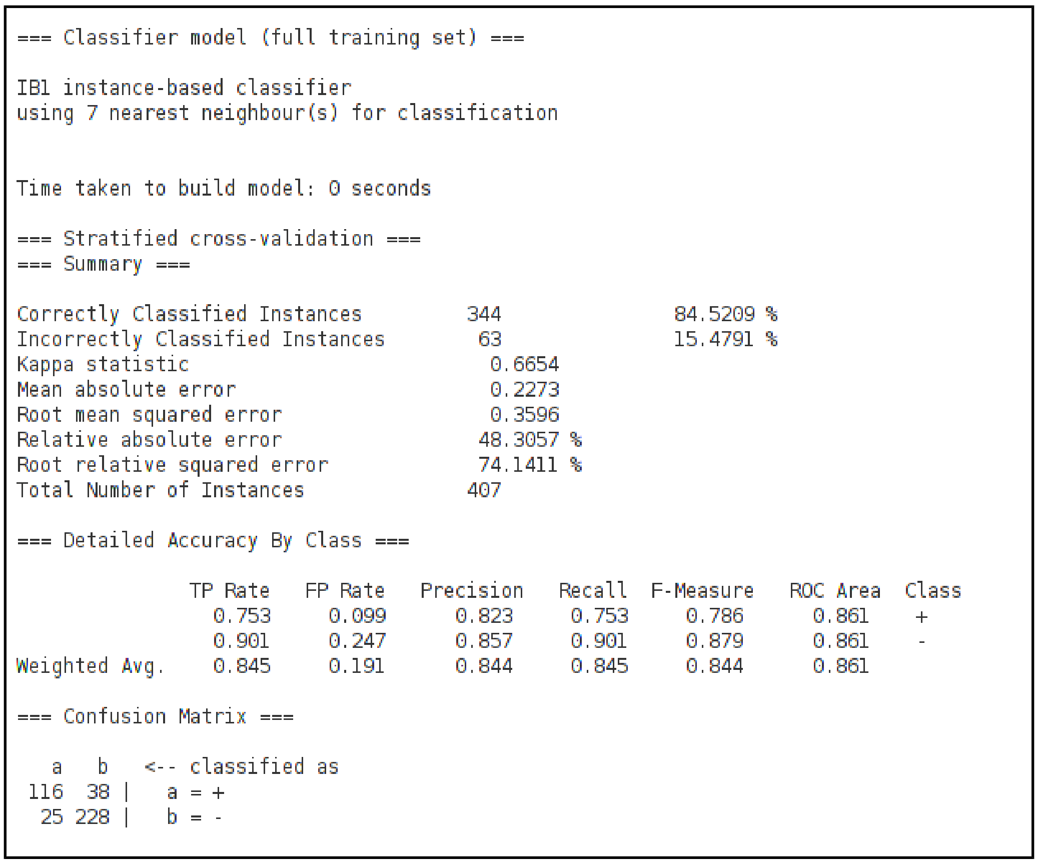
- **3-fold:** 84.28 %
- **4-fold:** 84.08 %
- **5-fold:** 84.52 %
- **6-fold:** 83.78 %
- **7-fold:** 84.52 %
- **8-fold:** 84.77 %
- **9-fold:** 84.28 %
- **10-fold:** 84.52 %

2. Second Algorithm: K Nearest Neighbors

The idea of the algorithm is simple, we are simply looking for the nearest points of our non-classified point.

In K Nearest Neighbors Classification, the distance between each instance from test-set and each instance from training-set is so important. For each test-set instance, nearest k training-set instances are chosen. Afterwards, first k instances' class labels determine the class label of the test-set instance's class label. It is recommended that, outliers should be eliminated before the operation, because if we choose k value too low then we can encounter with some outliers and predict the value wrong. Manhattan and Euclidean distance measures can be used to find distance between points. Manhattan distance is the simplest one and its performance is much higher than the other metrics.

In WEKA tool, K Nearest Neighbors Classification is present with name "weka.classifiers.lazy.IBk". In WEKA, k-fold cross validation method is used to determine training and test sets. Distance measure is **Manhattan Distance** because of its simplicity. Different k values are selected for KNN. The best value is **K=7 (7-NN)** for KNN. Also, different k-fold values are selected for k-fold cross validation method. For **k = 7 (7-fold)** we had the best results with accuracy rate **84.52 %**. Result is shown below:



```
=== Classifier model (full training set) ===

IB1 instance-based classifier
using 7 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         344               84.5209 %
Incorrectly Classified Instances        63               15.4791 %
Kappa statistic                          0.6654
Mean absolute error                      0.2273
Root mean squared error                  0.3596
Relative absolute error                 48.3057 %
Root relative squared error             74.1411 %
Total Number of Instances              407

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.753     0.099      0.823     0.753     0.786      0.861    +
                 0.901     0.247      0.857     0.901     0.879      0.861    -
Weighted Avg.    0.845     0.191      0.844     0.845     0.844      0.861

=== Confusion Matrix ===

   a   b   <-- classified as
 116  38 |   a = +
  25 228 |   b = -
```

According to results, **344 instances** are **correctly classified** and **63 instances** are **incorrectly classified**.

For other k-fold & k-NN values, accuracy rates are given below:

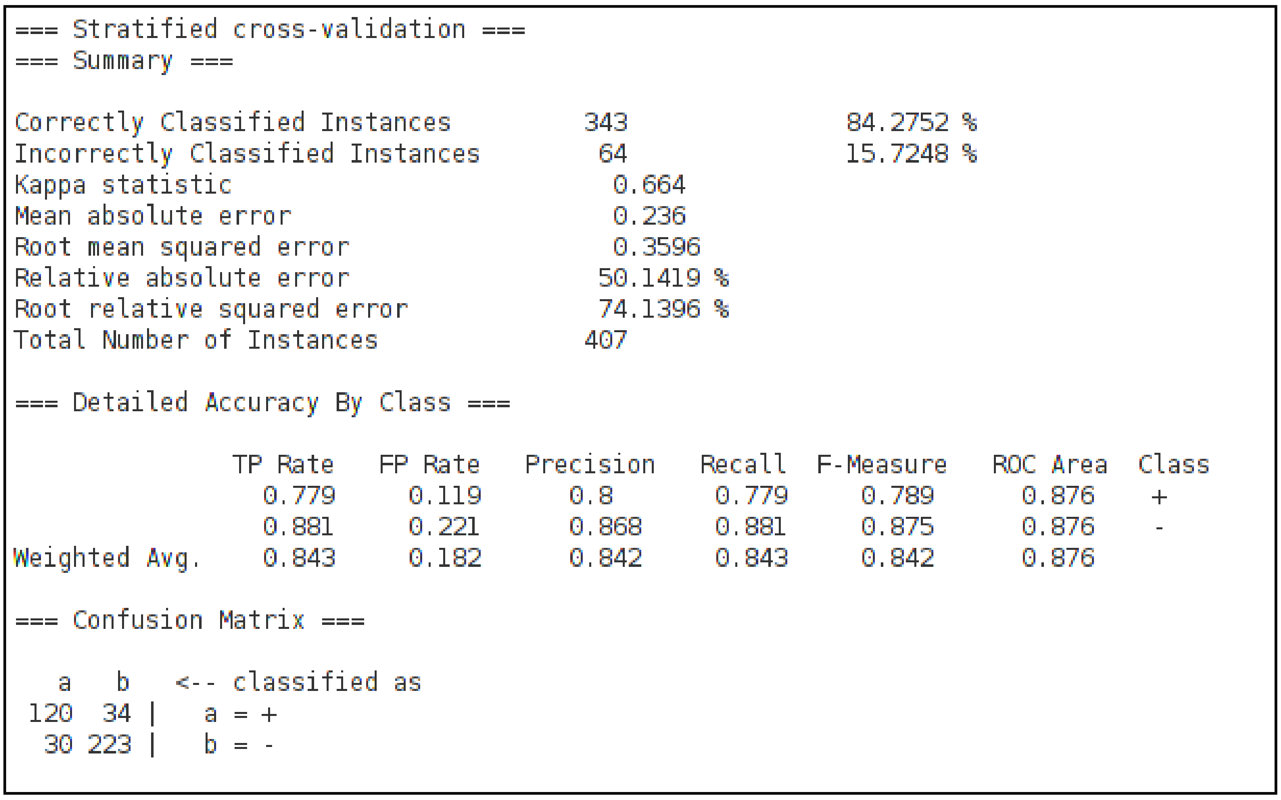
- **2-fold, 3-NN:** 82.31 %
- **3-fold, 3-NN:** 83.29 %
- **4-fold, 3-NN:** 82.31 %
- **5-fold, 3-NN:** 83.29 %
- **6-fold, 3-NN:** 82.80 %

- **7-fold, 3-NN:** 83.78 %
- **8-fold, 3-NN:** 83.54 %
- **9-fold, 3-NN:** 82.90 %
- **7-fold, 1-NN:** 79.61 %
- **7-fold, 5-NN:** 82.80 %
- **7-fold, 9-NN:** 84.03 %

3. Third Algorithm: Random Comittee

Class for constructing a tree that considers K randomly chosen attributes at each node. It performs no pruning. Also has an option to allow estimation of class probabilities based on a hold-out set (back fitting).

In WEKA tool, Random Comittee Classification is present with name "weka.classifiers.meta.RandomCommittee". In WEKA, k-fold cross validation method is used to determine training and test sets. Number of iterations is **10** and seed is **1.** Different k values are selected for k-fold cross validation method. For **k = 6 (6-fold)** we had the best results with accuracy rate **84.28 %** . Result is shown below:



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         343               84.2752 %
Incorrectly Classified Instances        64               15.7248 %
Kappa statistic                          0.664
Mean absolute error                      0.236
Root mean squared error                  0.3596
Relative absolute error                 50.1419 %
Root relative squared error             74.1396 %
Total Number of Instances              407

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.779     0.119      0.8        0.779     0.789      0.876    +
                 0.881     0.221      0.868      0.881     0.875      0.876    -
Weighted Avg.    0.843     0.182      0.842      0.843     0.842      0.876

=== Confusion Matrix ===

   a   b   <-- classified as
 120  34 |   a = +
  30 223 |   b = -
```

According to results, **343 instances** are **correctly classified** and **64 instances** are **incorrectly classified**.

For other k-fold values, accuracy rates are given below:

- **2-fold:** 81.82 %
- **3-fold:** 82.56 %
- **4-fold:** 82.31 %
- **5-fold:** 81.33 %
- **6-fold:** 84.28 %
- **7-fold:** 84.03 %
- **8-fold:** 79.85 %
- **9-fold:** 82.06 %
- **10-fold:** 82.31 %

1.c. Clustering

According to results of WEKA tool, the most successful clustering algorithms are respectively:

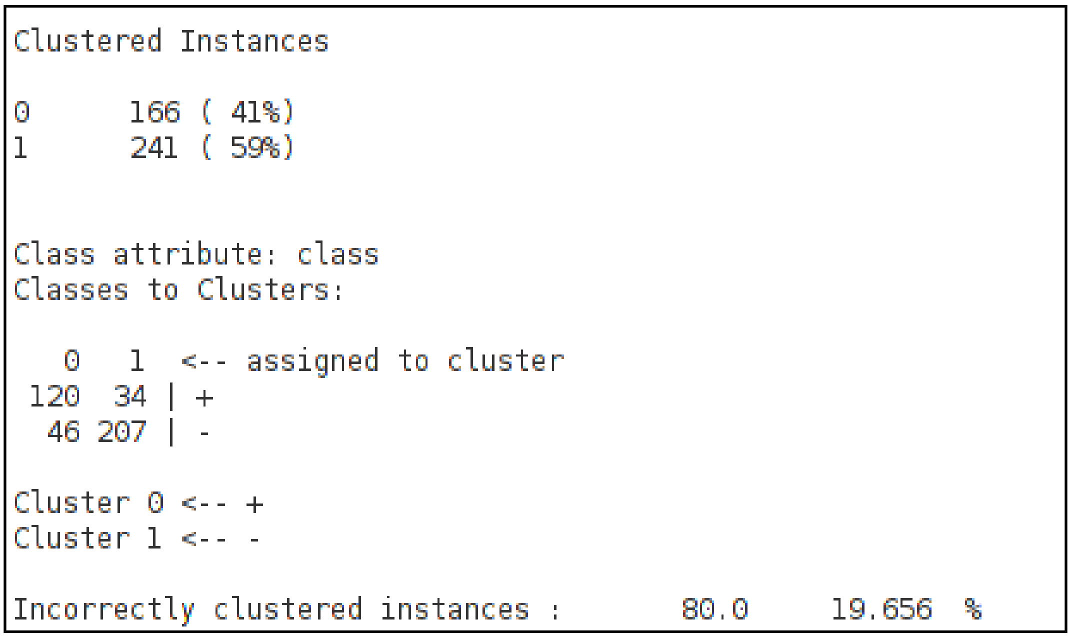
1. Simple K-Means
2. Farthest First
3. Hierarchical Clustering

Attribute “numOfCluster” is 2 for all three clustering algorithms, because we have only 2 class labels.

1. First Algorithm: Simple K-Means

We have randomly selected seeds at the beginning. Afterwards, the distance between all seeds and the dataset instances are calculated. If the distance between seed *S* and data-set instance *DS_I* is lower than the distance between other seeds and data-set instance *DS_I*, then this instance *DS_I* joins the seed *S*’s class. Because of the fact that the number of seeds (let’s say *N*) is equal to number of class labels, in each iteration we end up with *N* different clusters. At the end of each iterations, we calculate the mean of each cluster and each mean value becomes the next seed value. After a while, seed values and content of clusters no longer changes. Then we can say that we have reached the final version of clusters. Finally, we give class labels to final *N* clusters with majority voting.

In WEKA tool, Random Forest Classification is present with name "weka.clusterers.SimpleKMeans". Simple K Means algorithm's complexity is high. **Manhattan Distance** is selected as our distance metric, because there was no difference with the result which used Euclidean Distance. **Number of seeds** is **2**, because we have only 2 class labels. Accuracy rate is **80.34 % (100 – 19.565)**. Result is shown below:



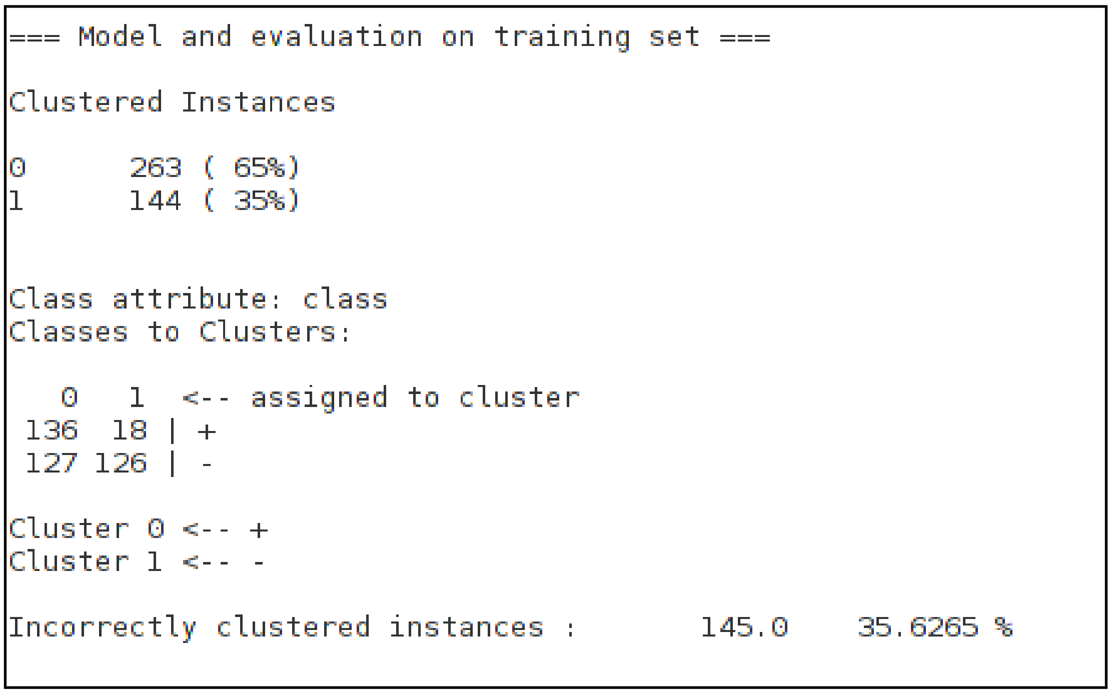
```
Clustered Instances

0      166 ( 41%)
1      241 ( 59%)


Class attribute: class
Classes to Clusters:

   0   1  <-- assigned to cluster
 120  34 | +
  46 207 | -

Cluster 0 <-- +
Cluster 1 <-- -

Incorrectly clustered instances :        80.0      19.656  %
```

According to results, **327 instances** are **correctly classified** and **80 instances** are **incorrectly classified**.

2. Second Algorithm: Farthest First

The farthest-first traversal of a bounded metric space is a sequence of points in the space, where the first point is selected arbitrarily and each successive point is as far as possible from the set of previously-selected points. The same concept can also be applied to a finite set of geometric points, by restricting the selected points to belong to the set or equivalently by considering the finite metric space generated by these points. For a finite metric space or finite set of geometric points, the resulting sequence forms a permutation of the points, known as the greedy permutation.

In WEKA tool, Farthest First Clustering is present with name "weka.clusterers.FarthestFirst". **Number of seeds** is **2**, because we have only 2 class labels. Accuracy rate is **64.37 % (100 – 35.6265)**. Result is shown below:



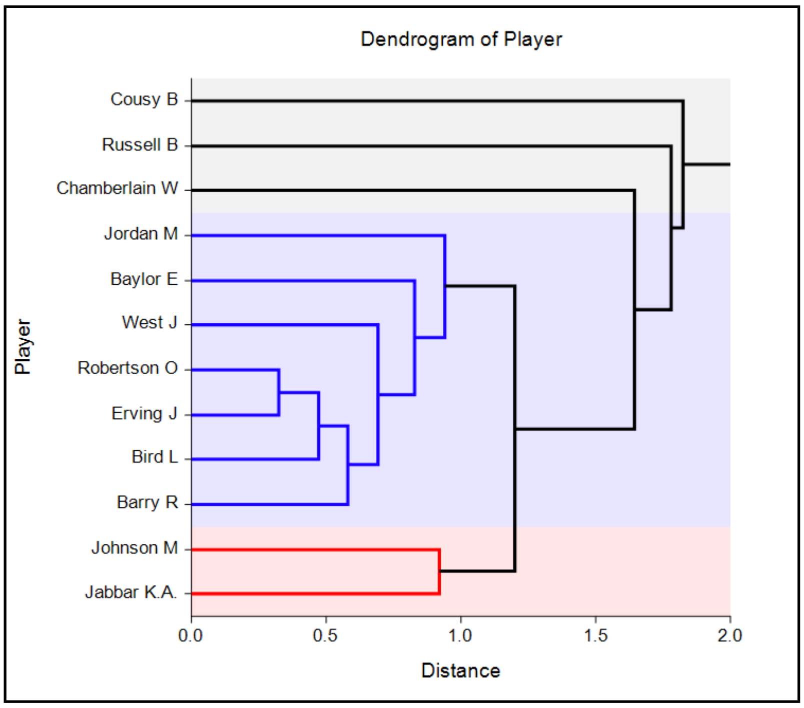
```
=== Model and evaluation on training set ===

Clustered Instances

0       263 ( 65%)
1       144 ( 35%)


Class attribute: class
Classes to Clusters:

   0   1  <-- assigned to cluster
 136  18 | +
 127 126 | -

Cluster 0 <-- +
Cluster 1 <-- -

Incorrectly clustered instances :       145.0    35.6265 %
```

According to results, **262 instances** are **correctly classified** and **145 instances** are **incorrectly classified**.

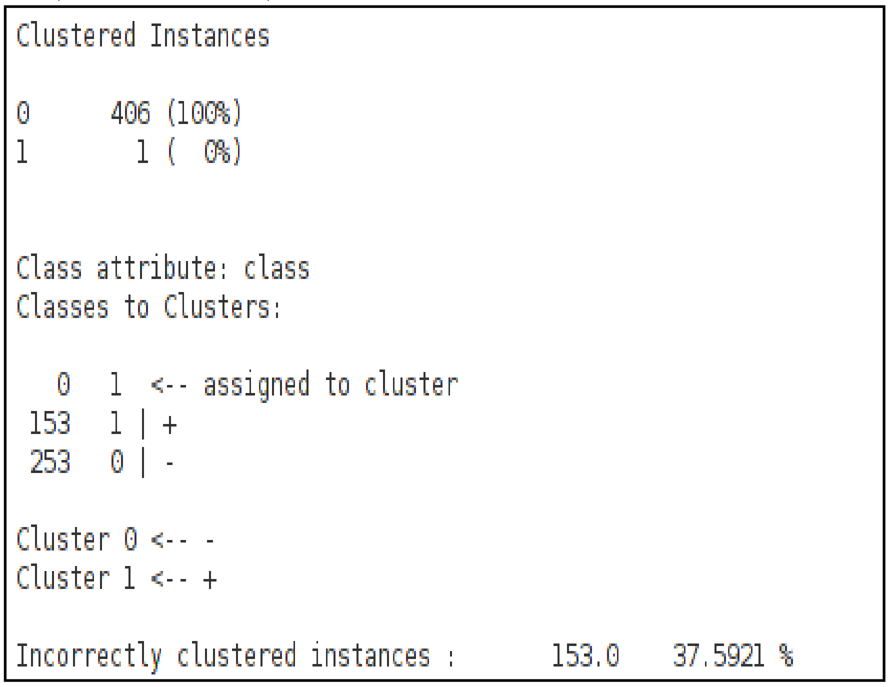
3. Third Algorithm: Hierarchical Clustering

Hierarchical Clustering is using dendrograms to represent clustering. There are 2 different hierarchical clustering methods. First one is agglomerative (bottom-up) method. Second one in top-down method. In agglomerative hierarchical clustering, each instance is cluster at the beginning. When the smallest distance was between instance A and instance B, instance A and instance B join the same cluster. Afterwards, if single linkage is being used then the second smallest distance is chosen in order to link the instances. After a while, cluster numbers become less and the distance between clusters become more. Finally, we can decide that each final cluster is a class. With majority voting, we can determine the class labels of each clusters.

An example of a dendrogram is shown below:



In WEKA tool, Hierarchical Clustering is present with name "weka.clusterers.HierarchicalClusterer". **Manhattan Distance** is selected as our distance metric, because there was no difference with the result which uses Euclidean Distance. Different link types are tested and for **single link,** we had the best results with accuracy rate **62.40 % (100 – 37.5921)**. Result is shown below:



```
Clustered Instances

0      406 (100%)
1        1 (  0%)


Class attribute: class
Classes to Clusters:

   0   1  <-- assigned to cluster
 153   1 | +
 253   0 | -

Cluster 0 <-- -
Cluster 1 <-- +

Incorrectly clustered instances :	153.0	 37.5921 %
```

According to results, **254 instances** are **correctly classified** and **153 instances** are **incorrectly classified**.

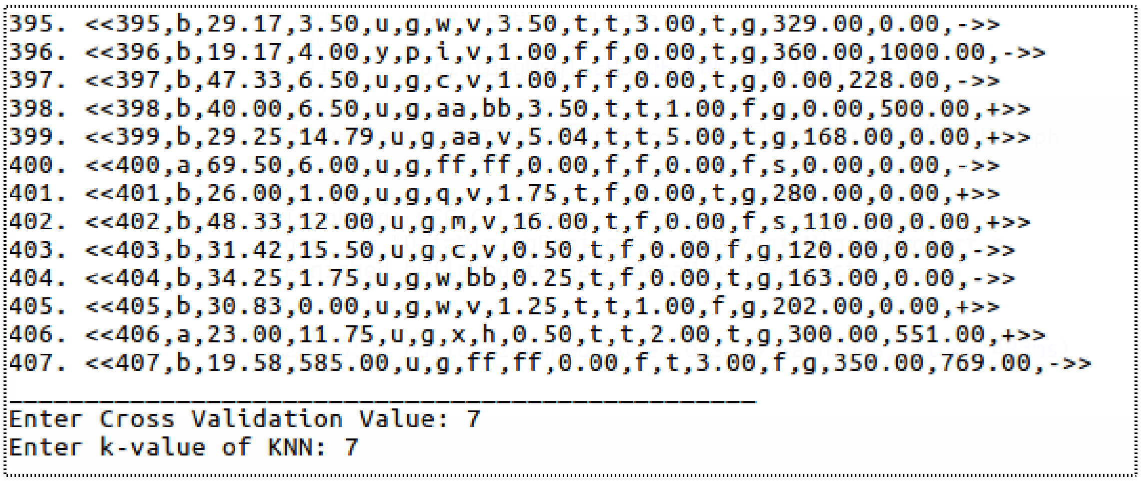
2. Code

Before the coding, the data-set was shuffled in order to make prediction more accurate.

2.a. KNN Classification

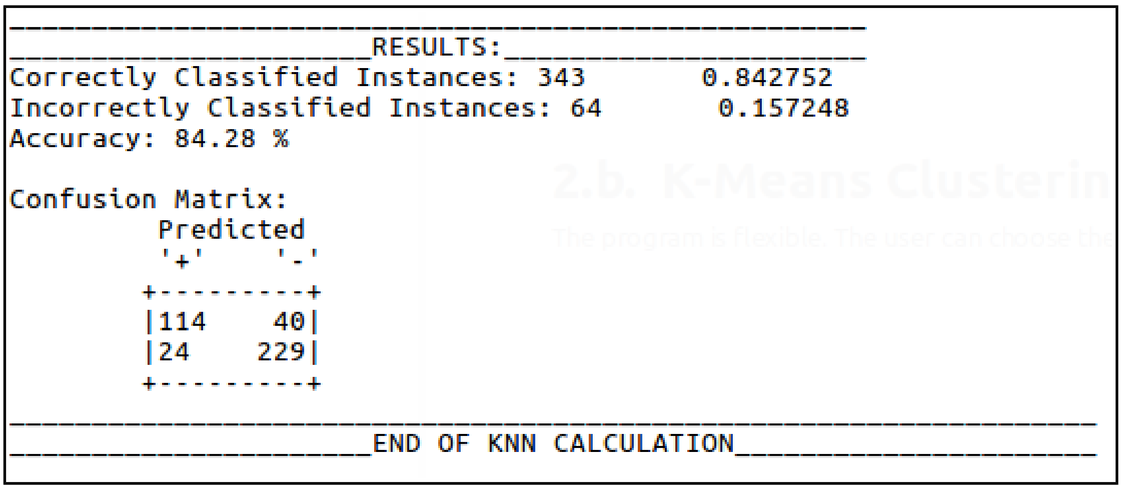
The program is flexible. The user can choose K-fold value for Cross Validation and K value for K-NN.

Entering Cross Validation Value and K-NN Value



```
395. <<395,b,29.17,3.50,u,g,w,v,3.50,t,t,3.00,t,g,329.00,0.00,->>
396. <<396,b,19.17,4.00,y,p,i,v,1.00,f,f,0.00,t,g,360.00,1000.00,->>
397. <<397,b,47.33,6.50,u,g,c,v,1.00,f,f,0.00,t,g,0.00,228.00,->>
398. <<398,b,40.00,6.50,u,g,aa,bb,3.50,t,t,1.00,f,g,0.00,500.00,+>>
399. <<399,b,29.25,14.79,u,g,aa,v,5.04,t,t,5.00,t,g,168.00,0.00,+>>
400. <<400,a,69.50,6.00,u,g,ff,ff,0.00,f,f,0.00,f,s,0.00,0.00,->>
401. <<401,b,26.00,1.00,u,g,q,v,1.75,t,f,0.00,t,g,280.00,0.00,+>>
402. <<402,b,48.33,12.00,u,g,m,v,16.00,t,f,0.00,f,s,110.00,0.00,+>>
403. <<403,b,31.42,15.50,u,g,c,v,0.50,t,f,0.00,f,g,120.00,0.00,->>
404. <<404,b,34.25,1.75,u,g,w,bb,0.25,t,f,0.00,t,g,163.00,0.00,->>
405. <<405,b,30.83,0.00,u,g,w,v,1.25,t,t,1.00,f,g,202.00,0.00,+>>
406. <<406,a,23.00,11.75,u,g,x,h,0.50,t,t,2.00,t,g,300.00,551.00,+>>
407. <<407,b,19.58,585.00,u,g,ff,ff,0.00,f,t,3.00,f,g,350.00,769.00,->>
_________________________________________________
Enter Cross Validation Value: 7
Enter k-value of KNN: 7
```

Result for 7-NN, 7-Fold & Confusion Matrix

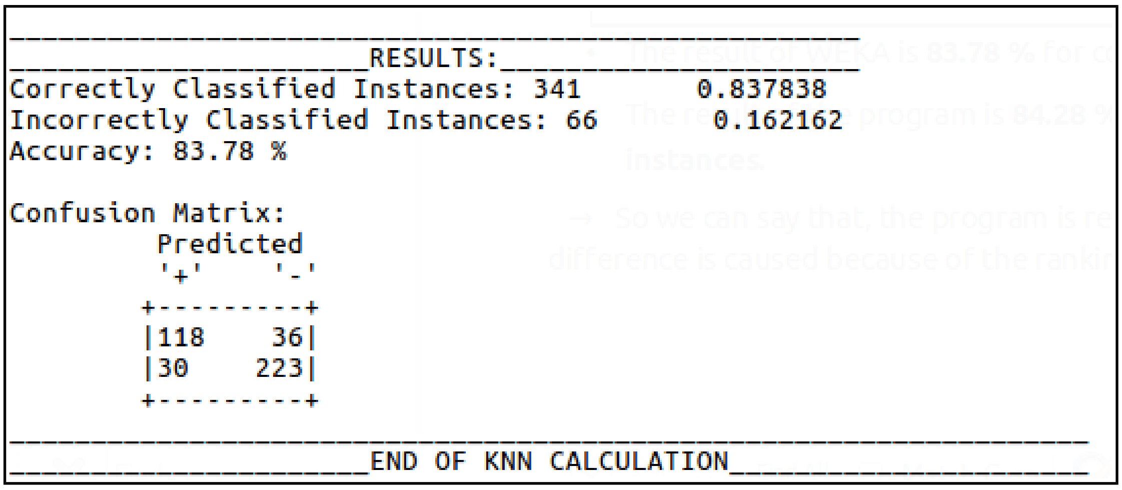


```
____________________________________________________
______________________RESULTS:______________________
Correctly Classified Instances: 343       0.842752
Incorrectly Classified Instances: 64       0.157248
Accuracy: 84.28 %

Confusion Matrix:
         Predicted
          '+'    '-'
        +---------+
        |114    40|
        |24    229|
        +---------+

__________________________________________________________________
______________________END OF KNN CALCULATION______________________
```

- The result of WEKA is **84.52 %** for correctly classified **344 instances**.
- The result of the program is **84.28 %** for correctly classified **343 instances.**

→ So we can say that, the program is really close to the WEKA. The reason of the difference is probably the ranking values for nominal values.

Result for 3-NN, 7-Fold & Confusion Matrix

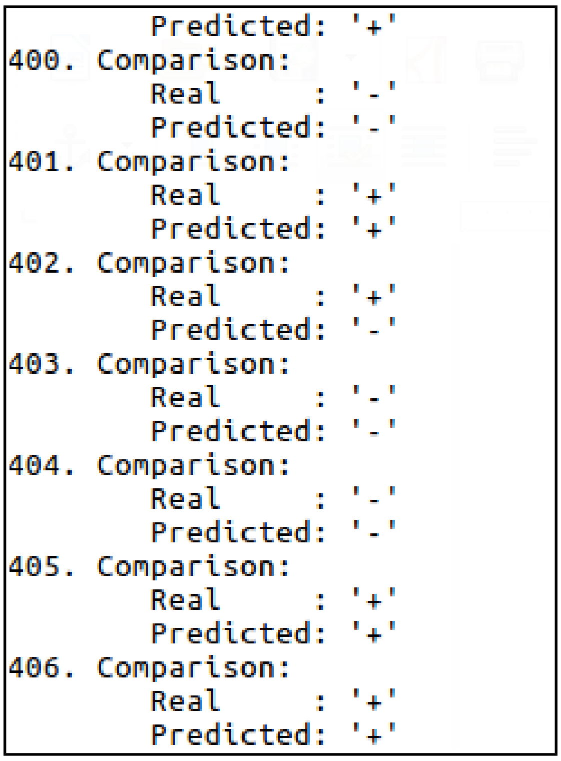


```
____________________________________________________
______________________RESULTS:______________________
Correctly Classified Instances: 341       0.837838
Incorrectly Classified Instances: 66       0.162162
Accuracy: 83.78 %

Confusion Matrix:
         Predicted
         '+'    '-'
        +---------+
        |118    36|
        |30    223|
        +---------+
__________________________________________________________________
______________________END OF KNN CALCULATION______________________
```

- The result of WEKA is **83.78 %** for correctly classified **341 instances**.
- The result of the program is **83.78 %** for correctly classified **341 instances.**

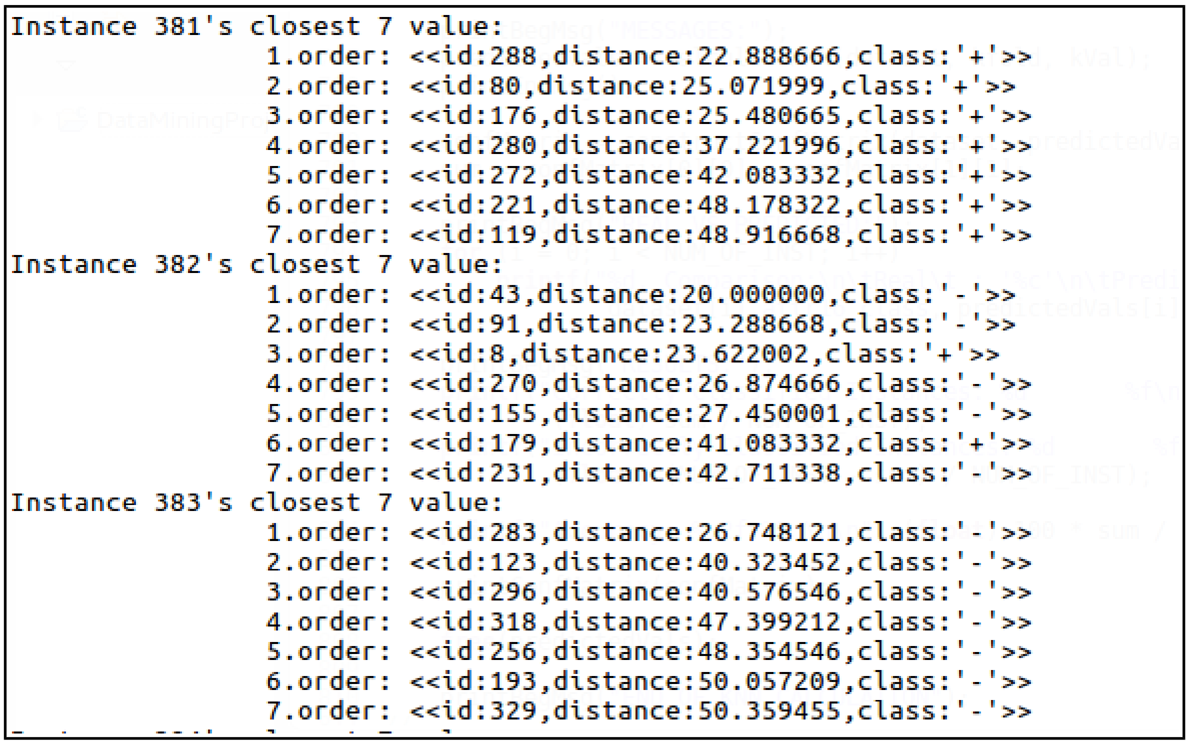
→ There is exact match.

Real vs. Predicted Value Comparison



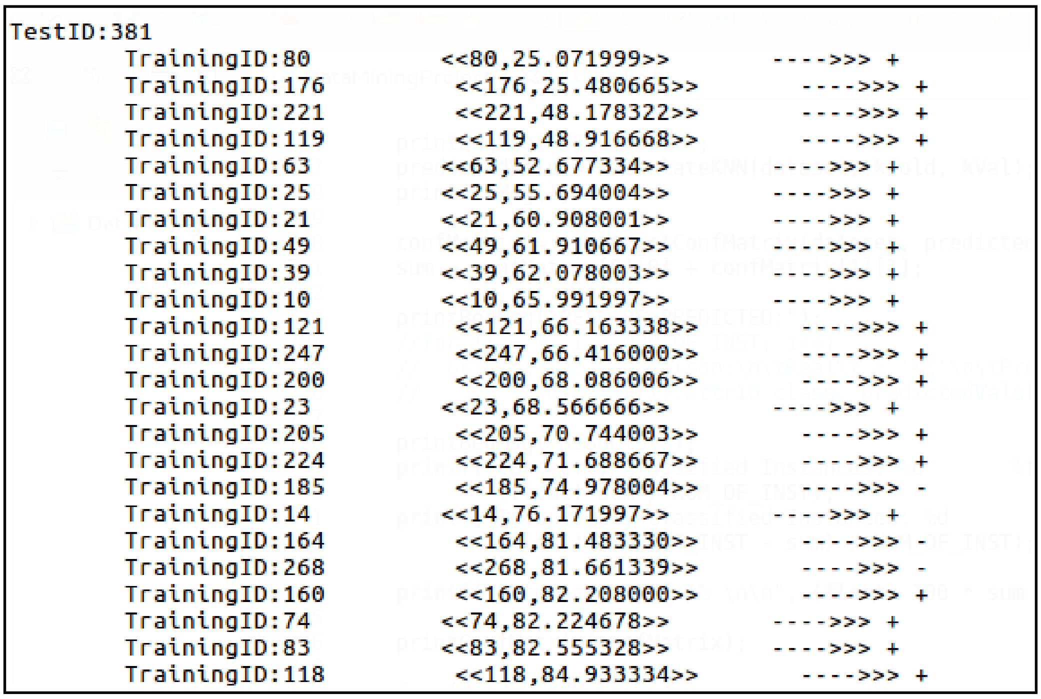
```
         Predicted: '+'
400. Comparison:
         Real     : '-'
         Predicted: '-'
401. Comparison:
         Real     : '+'
         Predicted: '+'
402. Comparison:
         Real     : '+'
         Predicted: '-'
403. Comparison:
         Real     : '-'
         Predicted: '-'
404. Comparison:
         Real     : '-'
         Predicted: '-'
405. Comparison:
         Real     : '+'
         Predicted: '+'
406. Comparison:
         Real     : '+'
         Predicted: '+'
```

First 7 Values for K=7 Output



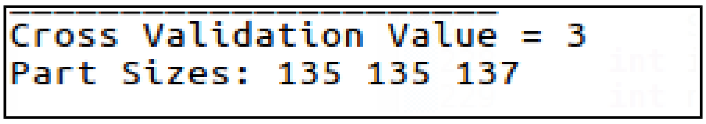
```
Instance 381's closest 7 value:
                1.order: <<id:288,distance:22.888666,class:'+'>>
                2.order: <<id:80,distance:25.071999,class:'+'>>
                3.order: <<id:176,distance:25.480665,class:'+'>>
                4.order: <<id:280,distance:37.221996,class:'+'>>
                5.order: <<id:272,distance:42.083332,class:'+'>>
                6.order: <<id:221,distance:48.178322,class:'+'>>
                7.order: <<id:119,distance:48.916668,class:'+'>>
Instance 382's closest 7 value:
                1.order: <<id:43,distance:20.000000,class:'-'>>
                2.order: <<id:91,distance:23.288668,class:'-'>>
                3.order: <<id:8,distance:23.622002,class:'+'>>
                4.order: <<id:270,distance:26.874666,class:'-'>>
                5.order: <<id:155,distance:27.450001,class:'-'>>
                6.order: <<id:179,distance:41.083332,class:'+'>>
                7.order: <<id:231,distance:42.711338,class:'-'>>
Instance 383's closest 7 value:
                1.order: <<id:283,distance:26.748121,class:'-'>>
                2.order: <<id:123,distance:40.323452,class:'-'>>
                3.order: <<id:296,distance:40.576546,class:'-'>>
                4.order: <<id:318,distance:47.399212,class:'-'>>
                5.order: <<id:256,distance:48.354546,class:'-'>>
                6.order: <<id:193,distance:50.057209,class:'-'>>
                7.order: <<id:329,distance:50.359455,class:'-'>>
```

Distance Between Test-set Instances and Training-set Instances



```
TestID:381
        TrainingID:80           <<80,25.071999>>        ---->>> +
        TrainingID:176           <<176,25.480665>>        ---->>> +
        TrainingID:221           <<221,48.178322>>        ---->>> +
        TrainingID:119           <<119,48.916668>>        ---->>> +
        TrainingID:63           <<63,52.677334>>        ---->>> +
        TrainingID:25           <<25,55.694004>>        ---->>> +
        TrainingID:21           <<21,60.908001>>        ---->>> +
        TrainingID:49           <<49,61.910667>>        ---->>> +
        TrainingID:39           <<39,62.078003>>        ---->>> +
        TrainingID:10           <<10,65.991997>>        ---->>> +
        TrainingID:121           <<121,66.163338>>        ---->>> +
        TrainingID:247           <<247,66.416000>>        ---->>> +
        TrainingID:200           <<200,68.086006>>        ---->>> +
        TrainingID:23           <<23,68.566666>>        ---->>> +
        TrainingID:205           <<205,70.744003>>        ---->>> +
        TrainingID:224           <<224,71.688667>>        ---->>> +
        TrainingID:185           <<185,74.978004>>        ---->>> -
        TrainingID:14           <<14,76.171997>>        ---->>> +
        TrainingID:164           <<164,81.483330>>        ---->>> -
        TrainingID:268           <<268,81.661339>>        ---->>> -
        TrainingID:160           <<160,82.208000>>        ---->>> +
        TrainingID:74           <<74,82.224678>>        ---->>> +
        TrainingID:83           <<83,82.555328>>        ---->>> +
        TrainingID:118           <<118,84.933334>>        ---->>> +
```

For 3-Fold Cross Validation, Size of Partitions Output

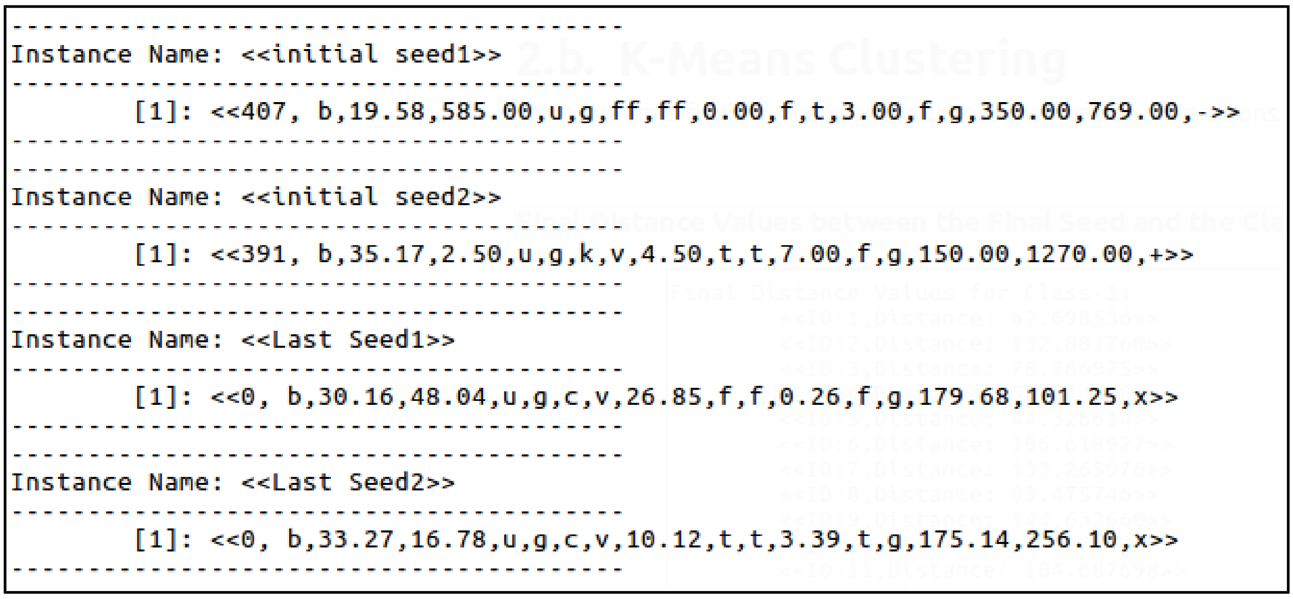


```
Cross Validation Value = 3
Part Sizes: 135 135 137
```

2.b. K-Means Clustering

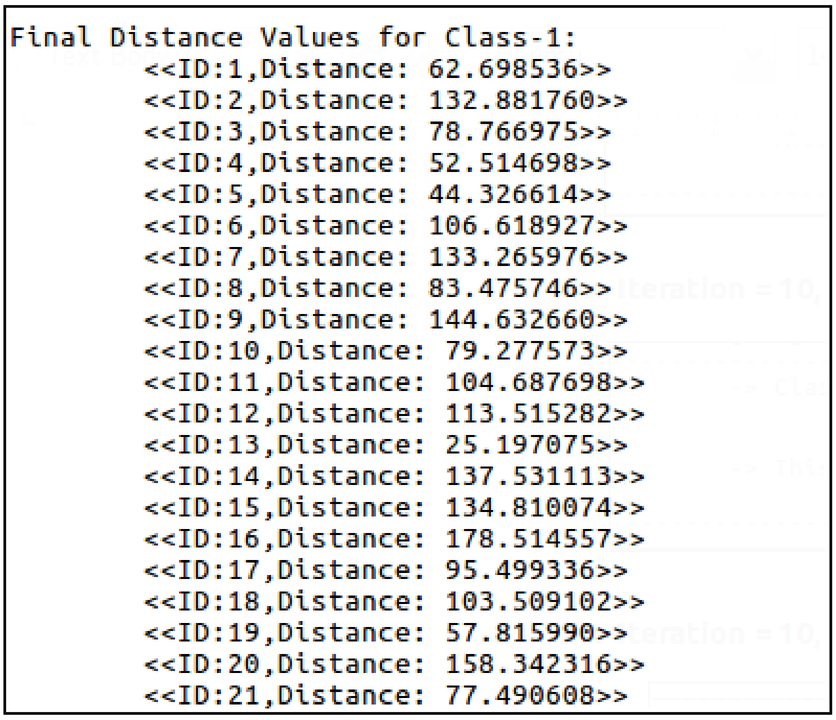
The program is flexible. The user can choose the number of iterations.

Initial Seeds & Final Seeds after Iterations



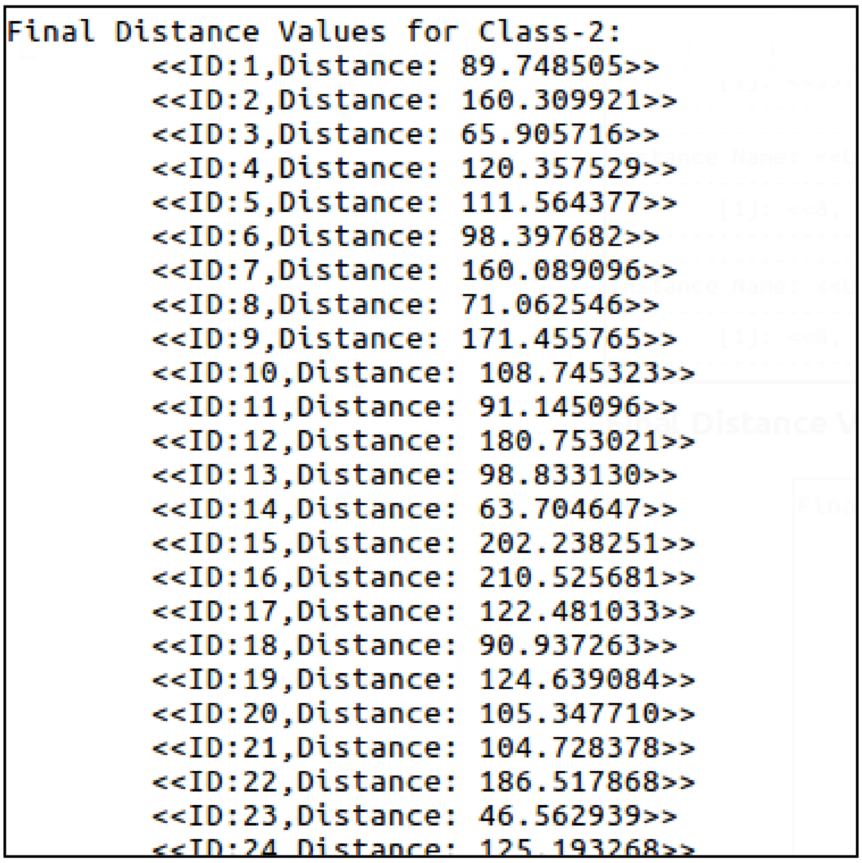
```
----------------------------------------
Instance Name: <<initial seed1>>
----------------------------------------
        [1]: <<407, b,19.58,585.00,u,g,ff,ff,0.00,f,t,3.00,f,g,350.00,769.00,->>
----------------------------------------
----------------------------------------
Instance Name: <<initial seed2>>
----------------------------------------
        [1]: <<391, b,35.17,2.50,u,g,k,v,4.50,t,t,7.00,f,g,150.00,1270.00,+>>
----------------------------------------
----------------------------------------
Instance Name: <<Last Seed1>>
----------------------------------------
        [1]: <<0, b,30.16,48.04,u,g,c,v,26.85,f,f,0.26,f,g,179.68,101.25,x>>
----------------------------------------
----------------------------------------
Instance Name: <<Last Seed2>>
----------------------------------------
        [1]: <<0, b,33.27,16.78,u,g,c,v,10.12,t,t,3.39,t,g,175.14,256.10,x>>
----------------------------------------
```

Final Distance Values between the Final Seed-1 and Class-1 Instances



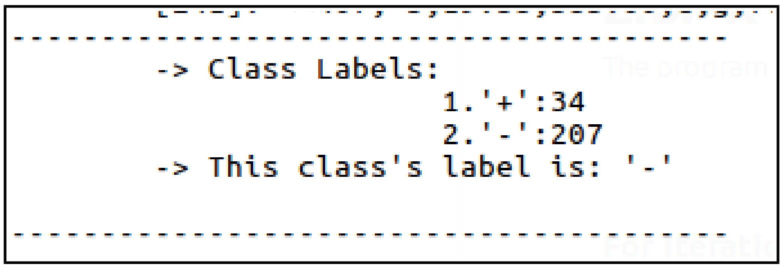
```
Final Distance Values for Class-1:
        <<ID:1,Distance: 62.698536>>
        <<ID:2,Distance: 132.881760>>
        <<ID:3,Distance: 78.766975>>
        <<ID:4,Distance: 52.514698>>
        <<ID:5,Distance: 44.326614>>
        <<ID:6,Distance: 106.618927>>
        <<ID:7,Distance: 133.265976>>
        <<ID:8,Distance: 83.475746>>
        <<ID:9,Distance: 144.632660>>
        <<ID:10,Distance: 79.277573>>
        <<ID:11,Distance: 104.687698>>
        <<ID:12,Distance: 113.515282>>
        <<ID:13,Distance: 25.197075>>
        <<ID:14,Distance: 137.531113>>
        <<ID:15,Distance: 134.810074>>
        <<ID:16,Distance: 178.514557>>
        <<ID:17,Distance: 95.499336>>
        <<ID:18,Distance: 103.509102>>
        <<ID:19,Distance: 57.815990>>
        <<ID:20,Distance: 158.342316>>
        <<ID:21,Distance: 77.490608>>
```

Final Distance Values between the Final Seed-2 and Class-2 Instances



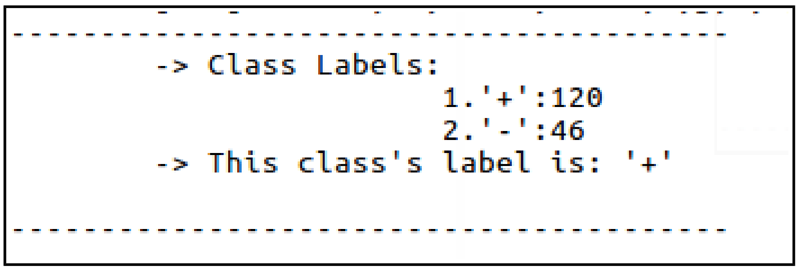
```
Final Distance Values for Class-2:
        <<ID:1,Distance: 89.748505>>
        <<ID:2,Distance: 160.309921>>
        <<ID:3,Distance: 65.905716>>
        <<ID:4,Distance: 120.357529>>
        <<ID:5,Distance: 111.564377>>
        <<ID:6,Distance: 98.397682>>
        <<ID:7,Distance: 160.089096>>
        <<ID:8,Distance: 71.062546>>
        <<ID:9,Distance: 171.455765>>
        <<ID:10,Distance: 108.745323>>
        <<ID:11,Distance: 91.145096>>
        <<ID:12,Distance: 180.753021>>
        <<ID:13,Distance: 98.833130>>
        <<ID:14,Distance: 63.704647>>
        <<ID:15,Distance: 202.238251>>
        <<ID:16,Distance: 210.525681>>
        <<ID:17,Distance: 122.481033>>
        <<ID:18,Distance: 90.937263>>
        <<ID:19,Distance: 124.639084>>
        <<ID:20,Distance: 105.347710>>
        <<ID:21,Distance: 104.728378>>
        <<ID:22,Distance: 186.517868>>
        <<ID:23,Distance: 46.562939>>
        <<ID:24,Distance: 125.193268>>
```

For Iteration = 10, Negative Class Label



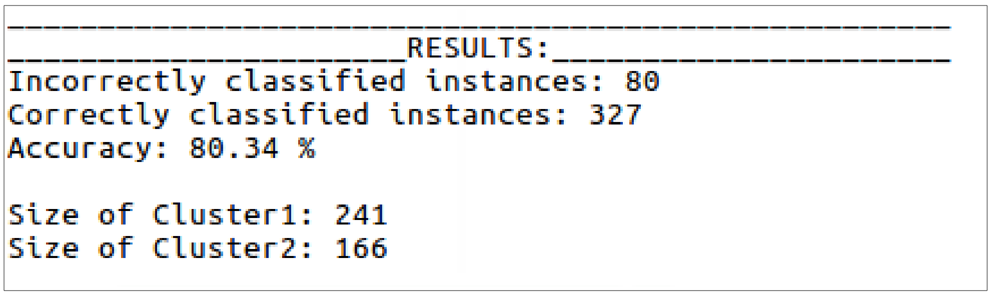
```
----------------------------------------
        -> Class Labels:
                        1.'+':34
                        2.'-':207
        -> This class's label is: '-'

----------------------------------------
```

For Iteration = 10, Positive Class Label



```
----------------------------------------
        -> Class Labels:
                        1.'+':120
                        2.'-':46
        -> This class's label is: '+'

----------------------------------------
```

For Iteration = 10, All Results



```
____________________________________________________
______________________RESULTS:______________________
Incorrectly classified instances: 80
Correctly classified instances: 327
Accuracy: 80.34 %

Size of Cluster1: 241
Size of Cluster2: 166
```

- The result of WEKA is **80.34 %** for correctly classified **327 instances**.
- The result of the program is **80.34 %** for correctly classified **327 instances.**

→ There is exact match. So we can say that, the program is really close to WEKA tool.

3. References

1. **WEKA Tool & Documentation:** Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.
2. **Dataset & Informations about the Dataset:** 'Evaluation of Features for Leaf Discrimination', Pedro F.B. Silva, Andre R.S. Marcal, Rubim M. Almeida da Silva (2013). Springer Lecture Notes in Computer Science, Vol. 7950, 197-204.
3. **Dataset Link:** https://archive.ics.uci.edu/ml/datasets/credit+approval
4. **Information about Random Forest Algorithm:** https://www.datascience.com/resources/notebooks/random-forest-intro
5. **Information about Density-Based Clustering Algorithm:** https://cse.buffalo.edu/~jing/cse601/fa12/materials/clustering_density.pdf
6. **Hierarchical Clustering Image:** https://ncss-wpengine.netdna-ssl.com/wp-content/uploads/2013/08/Hierarchical-Clustering-Dendrogram.png
7. **Information about Farthest First Algorithm**: https://en.wikipedia.org/wiki/Farthest-first_traversal