American International University-Bangladesh (AIUB)

Department of Computer Science

Faculty of Science &Technology (FST)

Summer 21-22

Section: A

Data Warehouse and Data Mining

PROJECT

A Report submitted
By

SN	Student Name	Student ID
1	SAJIDUL HASAN	18-38627-2
2	TURSHIN ARA ASHTARY	18-38593-2
3	ABDULLAH AL MAKSUD	18-38582-2

Under the supervision of

Tohedul Islam
Assistant Professor

Table of Contents

TASK 1:

Here is Our Dataset (Supervised Learning Dataset)

Introduction: Data mining using labelled data is known as supervised learning .where labelled data is a specially designated attribute and the aim is to use the data given to predict the value of that attribute for instance that has not to get been seen.

Targeted Features are: Housing In London Monthly Variable

Attributes: 7 attributes in this dataset

1. date
2. area
3. average_price
4. code

5. houses_sold
6. no_of_crimes
7. borough_flag

There is a total of 13549 instances of these 7 attributes and all these instances were used for classification. Here are the graphical details of the attributes:

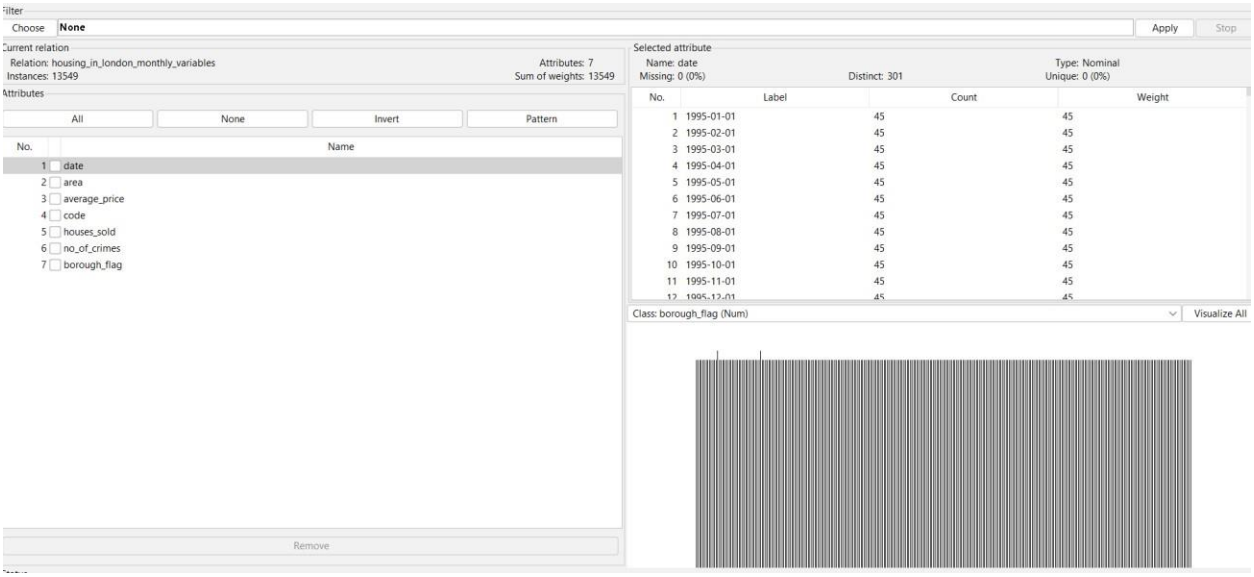


Fig 1: Selected dataset

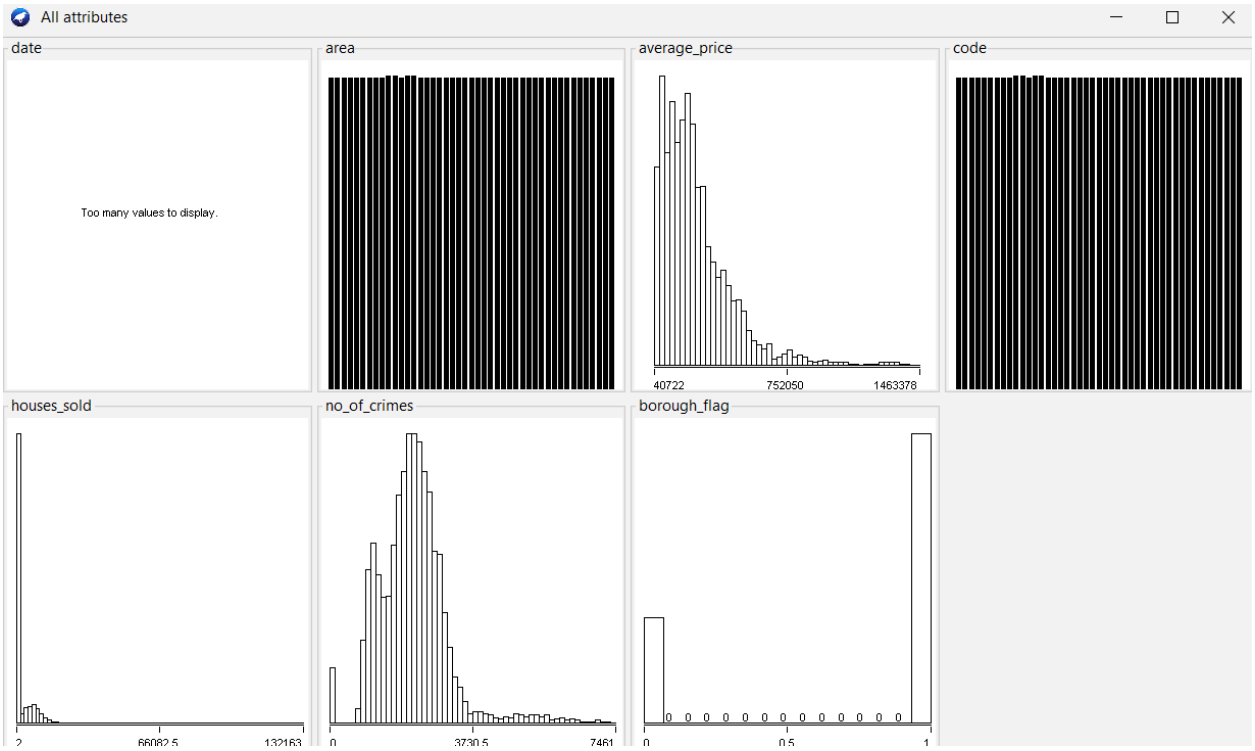


Fig 2: Details of all attribute

Applying Naïve Bayes Theorom: Naive Bayes algorithm is a supervised learning algorithm, which is based on the Bayes theorem and is used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. While classifying the selected dataset, the Naïve Bayes format was selected from the Bayes folder.

```
=== Run information ===

Scheme:       weka.classifiers.bayes.NaiveBayes
Relation:     housing_in_london_monthly_variables
Instances:    13549
Attributes:   7
              date
              area
              average_price
              code
              houses_sold
              no_of_crimes
              borough_flag
Test mode:    evaluate on training data

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

Naive Bayes Classifier
```

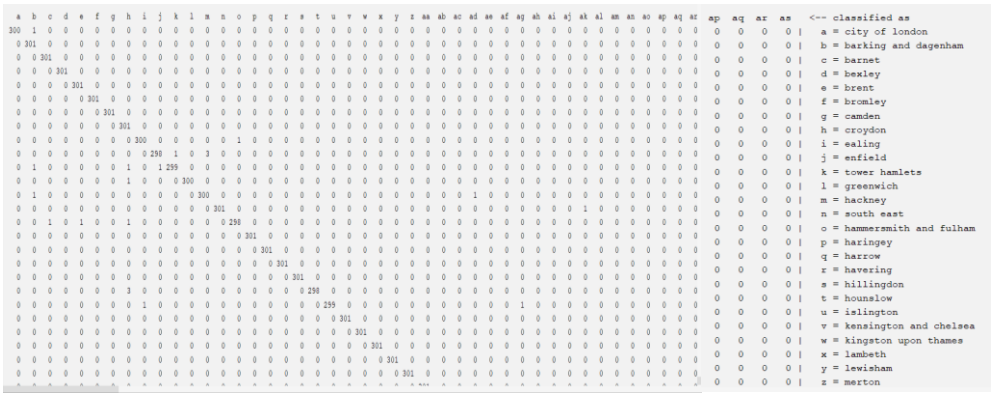
	Class						
Attribute	city of london (0.02)	barking and dagenham (0.02)	barnet (0.02)	bexley (0.02)	brent (0.02)	bromley (0.02)	camder (0.02)
date							
1995-01-01	2.0	2.0	2.0	2.0	2.0	2.0	2.
1995-02-01	2.0	2.0	2.0	2.0	2.0	2.0	2.
1995-03-01	2.0	2.0	2.0	2.0	2.0	2.0	2.
1995-04-01	2.0	2.0	2.0	2.0	2.0	2.0	2.
1995-05-01	2.0	2.0	2.0	2.0	2.0	2.0	2.
1995-06-01	2.0	2.0	2.0	2.0	2.0	2.0	2.
1995-07-01	2.0	2.0	2.0	2.0	2.0	2.0	2.
1995-08-01	2.0	2.0	2.0	2.0	2.0	2.0	2.
1995-09-01	2.0	2.0	2.0	2.0	2.0	2.0	2.

Fig-3

```
=== Evaluation on training set ===

Time taken to test model on training data: 8.2 seconds

=== Summary ===

Correctly Classified Instances       13537               99.9114 %
Incorrectly Classified Instances        12                0.0886 %
Kappa statistic                          0.9991
Mean absolute error                      0.002
Root mean squared error                  0.013
Relative absolute error                  4.4907 %
Root relative squared error              8.8265 %
Total Number of Instances            13549

=== Detailed Accuracy By Class ===
```

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	city of london
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	barking and dagenham
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	barnet
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	bexley
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	brent
1.000	0.000	0.997	1.000	0.998	0.998	1.000	1.000	bromley
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	camden
1.000	0.000	0.984	1.000	0.992	0.992	1.000	1.000	croydon
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	ealing
0.993	0.000	0.997	0.993	0.995	0.995	1.000	0.997	enfield
0.997	0.000	0.993	0.997	0.995	0.995	1.000	0.996	tower hamlets
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	greenwich
1.000	0.000	0.997	1.000	0.998	0.998	1.000	1.000	hackney

Fig-4: Applying Naïve Bayes Classifier



Applying KNN-Algorithm:

K-Nearest Neighbor classification usually works when all attributes are continuous, but it can be modified to deal with categorical attributes as well. The goal of KNN is to estimate the classification of an unseen

instance using the classification of the instance or instances that are nearest to it.

```
=== Evaluation on training set ===

Time taken to test model on training data: 22.66 seconds

=== Summary ===

Correctly Classified Instances       13545               99.9705 %
Incorrectly Classified Instances         4                0.0295 %
Kappa statistic                          0.9997
Mean absolute error                      0.0002
Root mean squared error                  0.0036
Relative absolute error                  0.3611 %
Root relative squared error              2.4755 %
Total Number of Instances            13549

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               1.000     0.000     1.000       1.000   1.000      1.000    1.000     1.000     E09000001
               1.000     0.000     1.000       1.000   1.000      1.000    1.000     1.000     E09000002
               1.000     0.000     1.000       1.000   1.000      1.000    1.000     1.000     E09000003
               1.000     0.000     1.000       1.000   1.000      1.000    1.000     1.000     E09000004
               1.000     0.000     1.000       1.000   1.000      1.000    1.000     1.000     E09000005
               1.000     0.000     1.000       1.000   1.000      1.000    1.000     1.000     E09000006
               1.000     0.000     1.000       1.000   1.000      1.000    1.000     1.000     E09000007
               1.000     0.000     1.000       1.000   1.000      1.000    1.000     1.000     E09000008
               1.000     0.000     1.000       1.000   1.000      1.000    1.000     1.000     E09000009
               0.997     0.000     0.997       0.997   0.997      0.997    0.998     0.993     E09000010
               0.997     0.000     0.997       0.997   0.997      0.997    0.998     0.993     E09000030
               1.000     0.000     1.000       1.000   1.000      1.000    1.000     1.000     E09000011
               0.997     0.000     0.997       0.997   0.997      0.997    0.998     0.993     E09000012
               0.997     0.000     0.997       0.997   0.997      0.997    0.998     0.993     E12000008
```

Fig-4: KNN Algorithm

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   s   t   u   v   w   x   y   z  aa  ab  ac  ad  ae  af  ag  ah  ai  aj  ak  al  am  an  ao  ap  aq  ar   as   <-- classified as
 180   0   0   0   0   0  41   0   2   0   0   0   0   0   0   0   0   0   0   2   0   1  26   0   0   0   0   0  11   0   0   0  38   0   0   0   0   0   0   0   0   0    0 |   a = city of london
   0 301   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   b = barking and dagenham
   0   0 288   0   0   0   0   0   1   0   1   0   0   0   0   7   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0   0   0   0   0   0   0   0    0 |   c = barnet
   0   0   0 299   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   d = bexley
   0   0   0   0 277   5   0   0   5   1   3   0   3   0   0   1   0   0   3   2   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   e = brent
   0   0   0   0   4 291   0   0   0   0   0   0   0   2   0   0   0   0   0   0   1   0   0   0   0   0   0   2   0   0   0   1   0   0   0   0   0   0   0   0   0   0    0 |   f = bromley
   0   0   0   0   0   0 301   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   g = camden
   0   0   0   0   0   0   0 289   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   5   0   4   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0    0 |   h = croydon
   0   0   0   0  11   0   0   0 269   0   4   0   3   0   0   8   0   0   0   0   0   0   0   3   0   0   0   0   0   3   0   0   0   0   0   0   0   0   0   0   0   0    0 |   i = ealing
   0   0   0   0   0   0   0   0   0 282   3   6   0   0   0   0   0   0   4   2   0   0   0   0   2   0   0   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   j = enfield
   0   0   0   0   3   0   0   0   8   1 277   0   6   0   0   3   0   0   0   1   0   0   0   0   0   0   0   2   0   1   0   0   0   0   0   0   0   0   0   0   0   0    0 |   k = tower hamlets
   0   0   0   0   0   0   0   3   0   3   0 271   0   0   0   0   0   0   3   3   0   0   0   0  11   0   0   2   0   0   0   5   0   0   0   0   0   0   0   0   0   0    0 |   l = greenwich
   0   0   1   0   4   3   0   0   2   0   4   0 278   1   0   2   0   0   0   0   0   0   0   2   0   0   1   0   0   4   0   0   0   0   0   0   0   0   0   0   0   0    0 |   m = hackney
   0   0   0   0   0   0   0   0   0   0   0   0   0 301   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   n = south east
   0   0   0   0   0   0   0   0   0   0   0   0   0   0 301   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   o = hammersmith and fulham
   0   0   7   0   1   0   0   0   9   0   3   0   3   0   0 278   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   p = haringey
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 296   0   0   0   0   0   2   0   0   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   q = harrow
   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0 299   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   r = havering
   0   0   0   0   3   0   0   0   0   0   3   1   1   0   0   0   0   0   0 268  11   0   0   0   1   3   0   0   8   0   0   0   2   0   0   0   0   0   0   0   0   0    0 |   s = hillingdon
   0   0   0   0   1   1   0   0   0   0   2   0   0   0   0   0   0   0   0  11 276   0   0   0   0   3   0   0   6   0   0   0   1   0   0   0   0   0   0   0   0   0    0 |   t = hounslow
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 297   0   0   0   0   0   0   0   0   0   0   4   0   0   0   0   0   0   0   0   0    0 |   u = islington
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 301   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   v = kensington and chelsea
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 301   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   w = kingston upon thames
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 295   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   x = lambeth
   0   0   0   0   0   0   0   0   6   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 278   0   0   0   0   0  17   0   0   0   0   0   0   0   0   0   0    0 |   y = lewisham
   0   0   0   0   0   0   0   3   0   5   0   6   0   0   0   0   0   0   5   3   0   0   0   0   0   0 267   0   1   2   0   0   9   0   0   0   0   0   0   0   0   0    0 |   z = merton
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0 297   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   aa = newham
   0   0   0   0   0   0   0   6   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0 294   0   0   0   0   0   0   0   0   0   0   0   0   0    0 |   ab = redbridge
   0   0   0   0   0   0   0   0   7   1   3   0   0   0   0   0   0   0   7   5   0   0   0   0   0   0   2   0   0 276   0   0   0   0   0   0   0   0   0   0   0   0    0 |   ac = richmond upon thames
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 301   0   0   0   0   0   0   0   0   0   0   0    0 |   ad = southwark
   0   0   0   0   1   0   0   0   2   0   0   0   3   0   0   0   0   0   0   0   0   0   0   0  12   0   0   0   0   0   0 283   0   0   0   0   0   0   0   0   0   0    0 |   ae = sutton
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 301   0   0   0   0   0   0   0   0   0    0 |   af = waltham forest
   0   0   0   0   0   1   0   1   0   3   0   8   0   0   0   0   0   0   6   4   0   0   0   0   8   0   0   1   0   0   0 269   0   0   0   0   0   0   0   0   0   0    0 |   ag = wandsworth
                                                                                                                                                                                    0 |   ah = westminster
```

RESULT:

Classifier	Accuracy
Naïve Bayes	99.91%
KNN	99.97%

Discussion:

After applying two types of classifiers, the highest percentage of correctly classified instances is for the naïve Bayes classifier with 99.9114%. After that comes the KNN classifier with 99.9705%. The KNN classifier is considered the best classifier for the dataset.

From the above discussion, we saw that the accuracy rate of the KNN classifier is higher than the Naïve Bayes

Task 2:

Introduction:

Supervised learning data(from task-1 Data set) to Test data set:

We make the test data set from the supervised learning data set we used before. We take here 2709 instances & 7 attribute in this data set.

Attribute are:

1. date
2. area
3. average_price
4. code
5. houses_sold
6. no_of_crimes
7. borough_flag

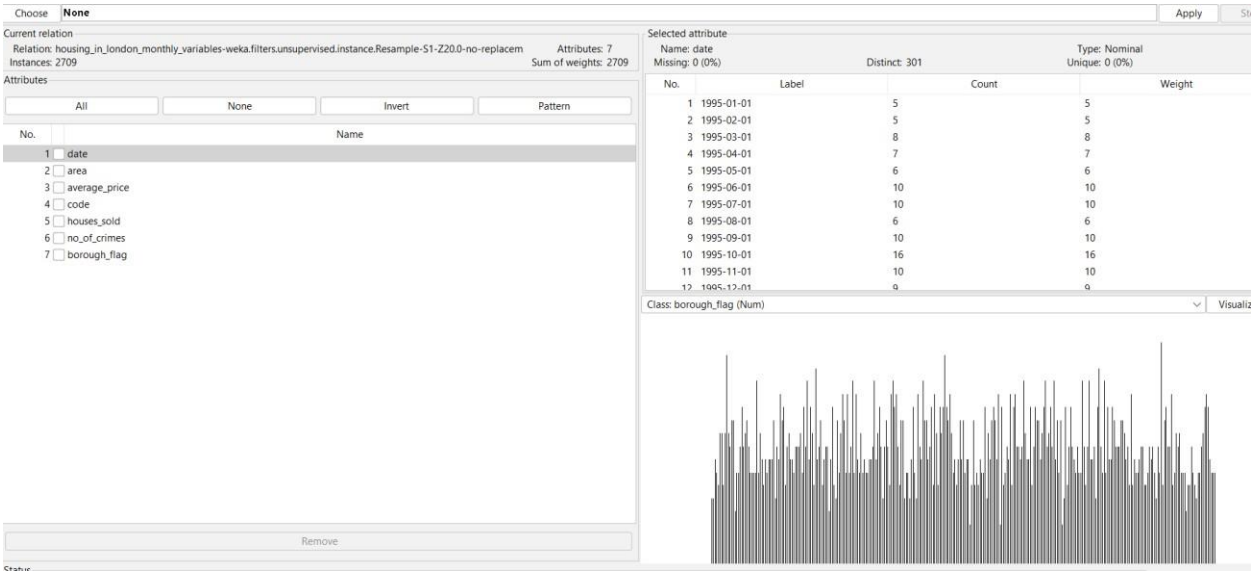


Fig 5: Selected dataset

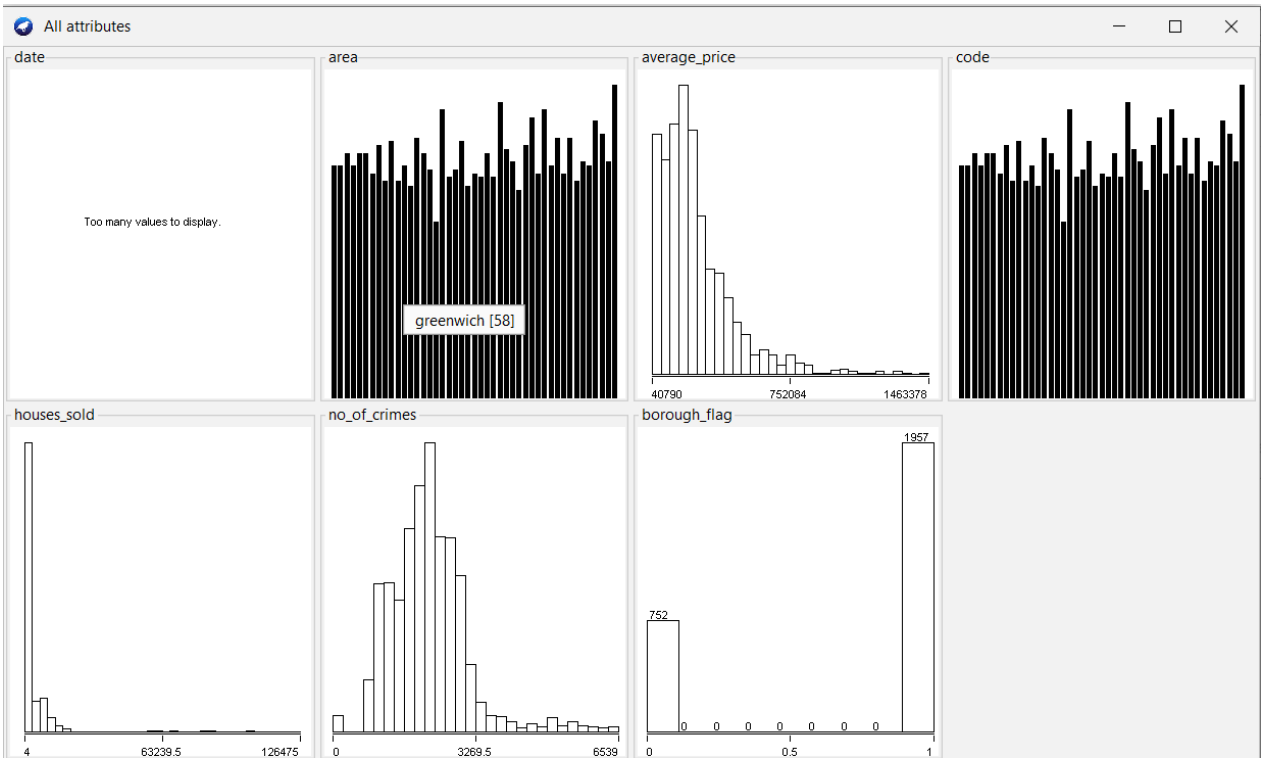
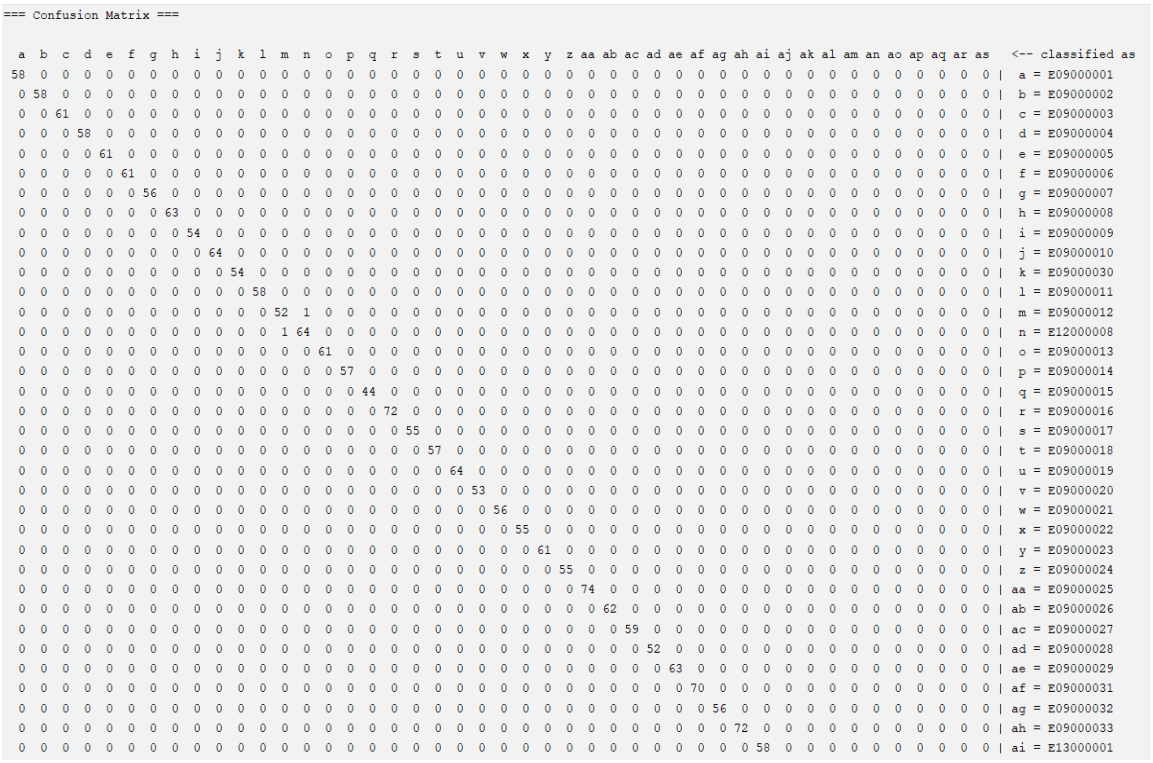


Fig 6: Details of all attribute

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 2.48 seconds

=== Summary ===

Correctly Classified Instances        2707               99.9262 %
Incorrectly Classified Instances         2                0.0738 %
Kappa statistic                          0.9992
Mean absolute error                      0
Root mean squared error                  0.0057
Relative absolute error                  0.1045 %
Root relative squared error              3.8757 %
Total Number of Instances             2709

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000001
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000002
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000003
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000004
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000005
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000006
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000007
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000008
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000009
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000010
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000030
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000011
               0.981    0.000    0.981      0.981    0.981      0.981    1.000     0.971     E09000012
               0.985    0.000    0.985      0.985    0.985      0.984    1.000     0.978     E12000008
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000013
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000014
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000015
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000016
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     E09000017
```

Fig 7: Applying J48 classifier



```
=== Confusion Matrix ===

 a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z aa ab ac ad ae af ag ah ai aj ak al am an ao ap aq ar as   <-- classified as
58  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  a = E09000001
 0 58  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  b = E09000002
 0  0 61  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  c = E09000003
 0  0  0 58  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  d = E09000004
 0  0  0  0 61  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  e = E09000005
 0  0  0  0  0 61  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  f = E09000006
 0  0  0  0  0  0 56  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  g = E09000007
 0  0  0  0  0  0  0 63  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  h = E09000008
 0  0  0  0  0  0  0  0 54  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  i = E09000009
 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  j = E09000010
 0  0  0  0  0  0  0  0  0  0 54  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  k = E09000030
 0  0  0  0  0  0  0  0  0  0  0 58  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  l = E09000011
 0  0  0  0  0  0  0  0  0  0  0  0 52  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  m = E09000012
 0  0  0  0  0  0  0  0  0  0  0  0  1 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  n = E12000008
 0  0  0  0  0  0  0  0  0  0  0  0  0  0 61  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  o = E09000013
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 57  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  p = E09000014
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 44  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  q = E09000015
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 72  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  r = E09000016
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 55  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  s = E09000017
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 57  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  t = E09000018
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  u = E09000019
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 53  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  v = E09000020
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 56  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  w = E09000021
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 55  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  x = E09000022
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 61  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  y = E09000023
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 55  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  z = E09000024
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 74  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | aa = E09000025
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 62  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | ab = E09000026
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 59  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | ac = E09000027
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 52  0  0  0  0  0  0  0  0  0  0  0  0  0 | ad = E09000028
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 63  0  0  0  0  0  0  0  0  0  0  0  0 | ae = E09000029
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 70  0  0  0  0  0  0  0  0  0  0  0 | af = E09000031
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 56  0  0  0  0  0  0  0  0  0  0 | ag = E09000032
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 72  0  0  0  0  0  0  0  0  0 | ah = E09000033
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 58  0  0  0  0  0  0  0  0 | ai = E13000001
```

Fig:Confusion Matrix

Result:

Classifier	Accuracy
Supervised Training Data Set (Using KNN classifier)	99.97%
Test Data Set(Using j48 classifier)	99.92%

Discussion:

The idea behind training and test sets is to test the generalization error. Separating data into training and testing sets is an important part of

evaluating data mining models. Since the test set data already contains known values for the property I want to predict, it is easy to determine if the model's assumptions are correct. So here I carefully separate the data into test/training and apply them in Weka after that it gives the error table and the accuracy rate is decreasing than before that's how I get my proper result and accuracy model.

I use here j48 classifier & it shows 99.62% accuracy whereas in the previous when we use it in the training dataset using KNN classifier it shows 99.97% accuracy.

TASk-3:

Introduction:

Unsupervised Learning is a machine learning technique in which the users do not need to supervise the model. Instead, it allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with unlabelled data. Unsupervised learning is helpful for finding useful insights from the data. Unsupervised learning is much similar to human learning to think by their own experiences, which makes it closer to the real AI. Unsupervised learning works on unlabeled and uncategorized data which makes unsupervised learning more important.

I have chosen the " accidental-deaths-in-USA-monthly" dataset. I will also use K means clustering Algorithm.

About the dataset:n this report, the used "accidental-deaths-in-usa-

monthly”, a CSV dataset file [Later converted into .arff], collected from

Kaggle.com.

RESULT :

Applying K means clustering Algorithm.

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on. It allows us to cluster the data into different groups and is a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

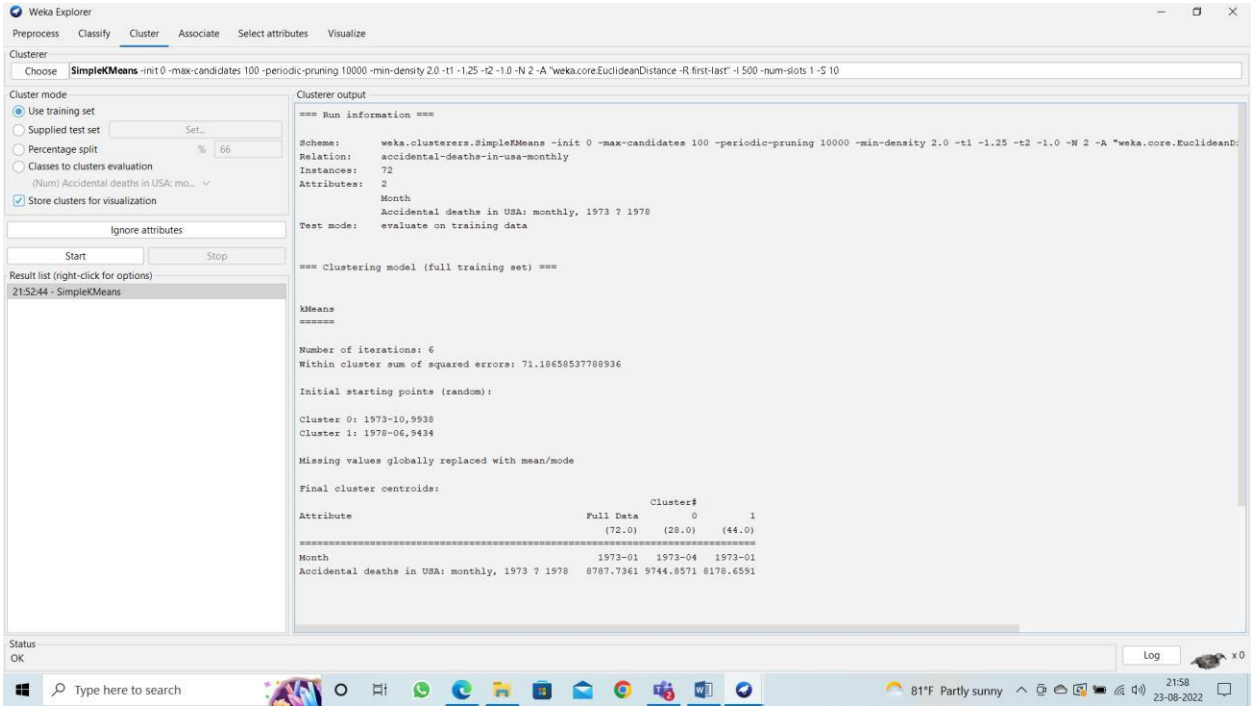


Fig-8

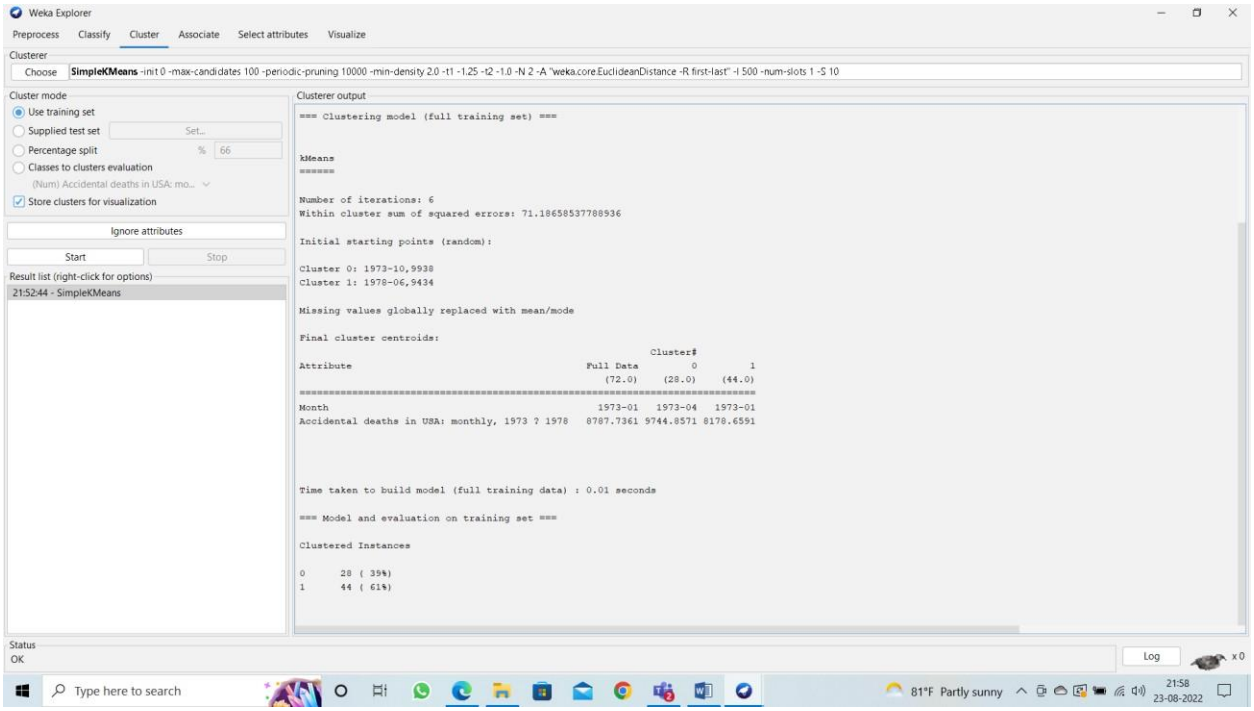


Fig-9

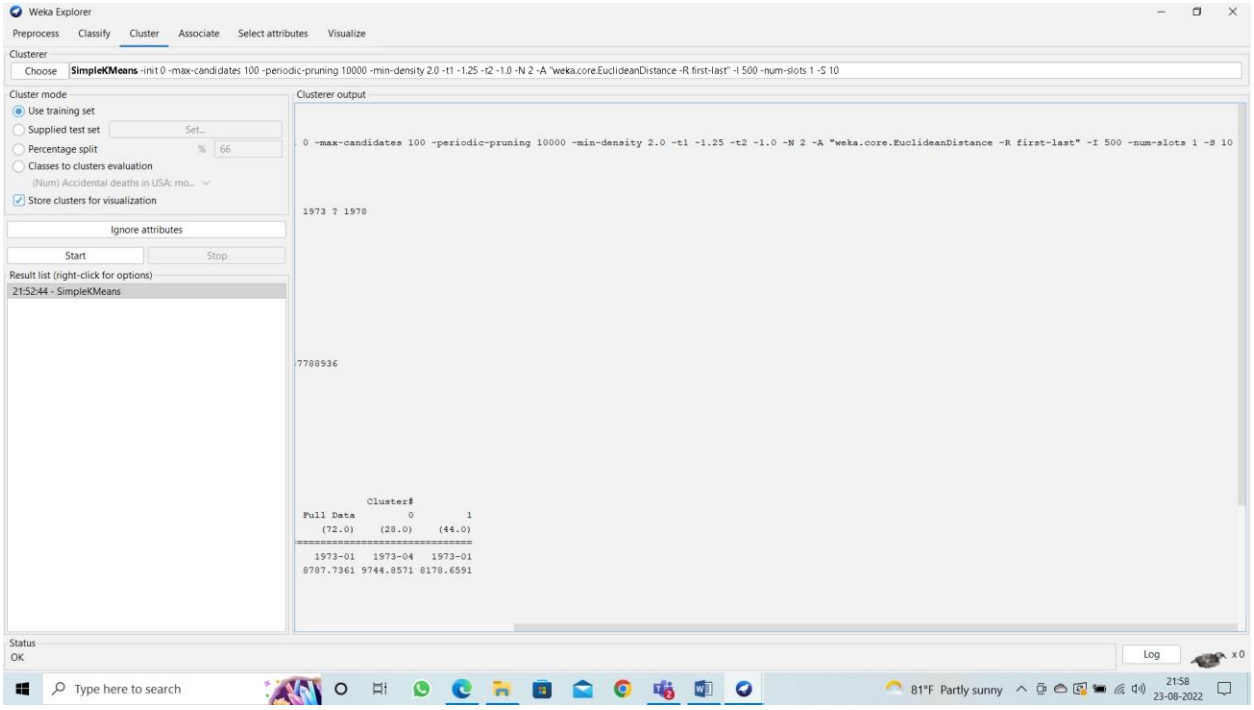


Fig-10

Figures 1,2,3,: K means clustering Algorithm

Here is the summary of the K means clustering Algorithm result:

Number of iterations: 6

Initial starting points (random):

Cluster 0: 1973-10,9938

Cluster 1: 1978-06,9434

Time taken to build model (full training data) : 0.01 seconds

Clustered Instances

0	28(39%)
1	44(61%)

Discussion:

I have chosen a proper unsupervised dataset. I convert csv file to arff file and then apply K means Clustering Algorithm. By default, the value of K is 2 so there are two results. For 0 instances it is 39% whereas for 1 is 61%. I also find the final cluster of centroids also. By that I have completed K means clustering for unsupervised data