CITS5504 Group Project 2

Data Mining

Harper Wu (23052765)
Isaac Huang (23019722)

Table of Contents

1. DATA CLEANING AND ANALYSIS

1.1 Irrelevant Data or Duplicates

1.1.1 Remove Attribute ID

IDs are not useful for data warehousing or data mining, so we removed this attribute.

1.1.2 Remove Attribute three_g

While building classifiers, when four_g = 1, all three_g = 1. This association had always been picked out. There was a very strong correlation between four_g and three_g, and theoretically speaking, four_g mobiles should be able to support three_g. Also, it is a better indicator for a high price mobile. Hence, we chose to remove three_g attribute.

1.2 Fix Structural Errors

In attributes “blue”, “dual_sim”, “wifi”, the data were not consistent, so we used the Excel function below to fix them:

```
=IFS(OR(C2={"no","NO","not","No","Not"}),0, OR(C2={"yes","YES","has","Has","Yes"}),1)
```

1.3 Missing or Extreme Values

1.3.1 Assumption

At this step, when fc = 0, we assume that mobile does not have a front camera.
At this step, when pc = 0, we assume that mobile does not have a primary camera.

1.3.2 Missing Data

Weka clearly showed missing data is 0% for all attributes.

1.3.3 Extreme and Non-sense Values

Some of the px_height values seem irrationally small, even with value 0.

From the lowest px_width = 500 and the highest sc_w = 18, we can obtain the lowest theoretical resolution rate applied among these mobiles is no less than “70 dpi” (Figure 1). Since our lowest sc_h = 5 cm, the px_height must be greater than 137.79 (Figure 2). Hence, we replaced all values from 0 to 137 in px_height with “?”, then applied “ReplaceMissingValues” function to replace them with the mean in Weka.

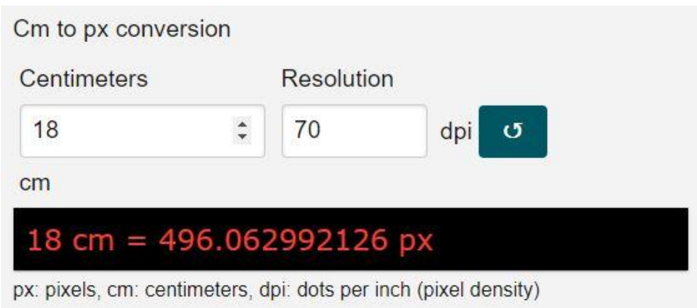


Figure 1. Lowest resolution calculated by lowest px_width and highest sc_w

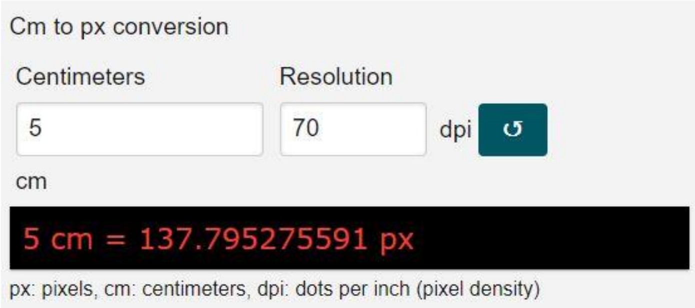


Figure 2. px_height calculated by lowest sc_h and resolution as 70 dpi

Some of the sc_w values seem impractically small, even with value 0. From the highest px_height = 1960 and the lowest sc_h = 5 cm, we can obtain the highest theoretical resolution rate applied among these mobiles is no higher than ”996 dpi” (Figure 3). Since our lowest px_width is 500, the sc_w must be greater than 1 cm (Figure 4). Hence, we replaced 0 and 1 in sc_w with “?”, then applied “ReplaceMissingValues” function to replace them with the mean in Weka.

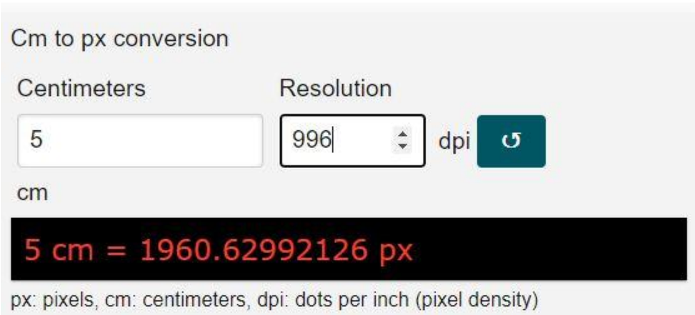


Figure 3. Highest theoretical resolution calculated by highest px_height and lowest sc_h

Figure 4. Lowest sc_w calculated by lowest px_width and higest theoretical resolution

1.3.4 Noise or Dirty Data

Even after we applied the last step, there were still a lot of data that did not make any sense. The ratio of pixels and the ratio of screen height to width are completely reversed. See examples in Table 1 below:

px_height	px_width	sc_h	sc_w
138	1371	13	6
142	1039	9	3
148	1606	19	8
150	1897	13	2
163	1011	15	2

Table 1. Examples of the ratio of pixels and the ratio of screen size

Attribute data of px_height and sc_w seemed unfitted and dirty, especially when we put them side by side with px_width and sc_w. They made a lot of noise and distraction during our analysis. Hence, we decided to remove px_height and sc_w, but keep px_width and sc_h.

Px_width and sc_h overall can still give us a general idea of how big or how clear the mobile screen would be.

1.4 Discretization

Association rule mining can only be performed on categorical data, so we have to perform discretization here, but it does not mean we have to use the same discrete data for training classifiers or running clustering algorithms in other tasks.

First, we needed to change our 0/1 numerical attribute data, "blue", "dual_sim", "wifi", "four_g", "touch_screen" and "price_category", to nominal {0, 1}.

```
@attribute battery_power numeric
@attribute blue {0,1}
@attribute clock_speed numeric
@attribute dual_sim {0,1}
@attribute fc numeric
@attribute four_g {0,1}
@attribute int_memory numeric
@attribute m_dep numeric
@attribute mobile_wt numeric
@attribute n_cores numeric
@attribute pc numeric
@attribute px_height numeric
@attribute px_width numeric
@attribute ram numeric
@attribute sc_h numeric
@attribute sc_w numeric
@attribute talk_time numeric
@attribute touch_screen {0,1}
@attribute wifi {0,1}
@attribute price_category {0,1}
```

Figure 5. Change numerical attributes to nominal attributes

We use 4 bins with equal frequency. The number of bins is based on our numerous experiments. We use equal frequency here, as the data is not evenly distributed. Some dataset could have 1462 instances while one of the others was only 125.

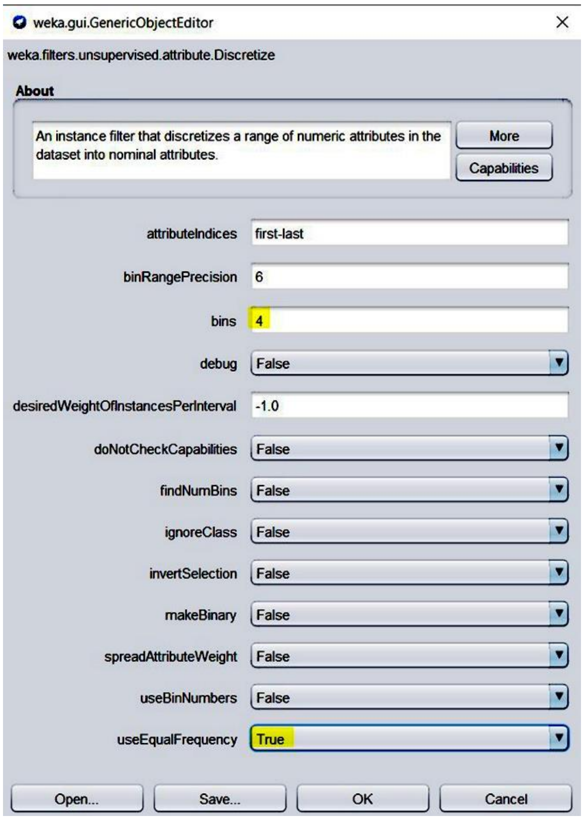


Figure 6. Discretise numerical attributes into 4 bins with equal frequency

2. ASSOCIATION RULE MINING

2.1 Association Rule - Apriori Algorithm

Here, we have to set up “support” and “confidence / lift” value.

In our data, we have 1,500 instances of low-price mobile data and 500 instances of high-price mobile data, so our support value must be under “500/2000 = 0.25” in order to mine interesting patterns for high price_category.

We started with a high support 0.25 and a high enough lift 1.4, and then we gradually decreased the support value until we found some interesting patterns. At support 0.11 and lift 1.4, we got some interesting results shown in the next step.

We used lift here, as results with confidence variables sometimes would not be so interesting.

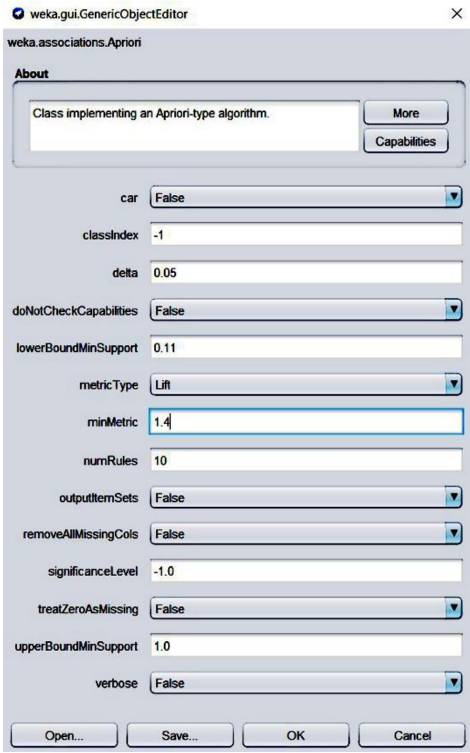


Figure 7. Set parameters for Apriori Algorithm

2.2 Result

We mined out our 10 top ranked rules that had the highest lift value, which had to be greater than the minimum value 1.4:

1. ram='(3065-inf)' 500 ==> dual_sim=1 price_category=1 230   conf:(0.46) < lift:(3.47)> lev:(0.08) [163] conv:(1.6)
2. dual_sim=1 price_category=1 265 ==> ram='(3065-inf)' 230   conf:(0.87) < lift:(3.47)> lev:(0.08) [163] conv:(5.52)
3. four_g=1 ram='(3065-inf)' 271 ==> price_category=1 232   conf:(0.86) < lift:(3.42)> lev:(0.08) [164] conv:(5.08)
4. price_category=1 500 ==> four_g=1 ram='(3065-inf)' 232   conf:(0.46) < lift:(3.42)> lev:(0.08) [164] conv:(1.61)
5. ram='(3065-inf)' 500 ==> four_g=1 price_category=1 232   conf:(0.46) < lift:(3.37)> lev:(0.08) [163] conv:(1.6)
6. four_g=1 price_category=1 275 ==> ram='(3065-inf)' 232   conf:(0.84) < lift:(3.37)> lev:(0.08) [163] conv:(4.69)
7. ram='(3065-inf)' 500 ==> price_category=1 417   conf:(0.83) < lift:(3.34)> lev:(0.15) [292] conv:(4.46)
8. price_category=1 500 ==> ram='(3065-inf)' 417   conf:(0.83) < lift:(3.34)> lev:(0.15) [292] conv:(4.46)
9. dual_sim=1 ram='(3065-inf)' 278 ==> price_category=1 230   conf:(0.83) < lift:(3.31)> lev:(0.08) [160] conv:(4.26)
10. price_category=1 500 ==> dual_sim=1 ram='(3065-inf)' 230   conf:(0.46) < lift:(3.31)> lev:(0.08) [160] conv:(1.59)

The three rules we are interested are as below:

3. four_g=1 ram='(3065-inf)' 271 ==> price_category=1 232   conf:(0.86) < lift:(3.42)> lev:(0.08) [164] conv:(5.08)

This means "232 mobiles are categorised as high-price mobile among 271 mobiles that support 4G wifi and whose RAM size is bigger than 3065 MB."

7. ram='(3065-inf)' 500 ==> price_category=1 417 conf:(0.83) < lift:(3.34)> lev:(0.15) [292] conv:(4.46)

This means “417 mobiles are categorised as high-price mobile among 500 mobiles whose RAM size is bigger than 3065 MB.”

9. dual_sim=1 ram='(3065-inf)' 278 ==> price_category=1 230 conf:(0.83) < lift:(3.31)> lev:(0.08) [160] conv:(4.26)

This means “230 mobiles are categorised as high-price mobile among 278 mobiles that support dual sim cards and whose RAM size is bigger than 3065 MB.”

2.3 Recommendation

From the three association rules above, our recommendation for a company to design a high-price mobile phone is that the RAM size MUST be bigger than 3065 MB, and it could be better if it supports 4G Wi-Fi network. Otherwise, to be able to support dual sim cards may be another good option too.

3. CLASSIFICATION

3.1 Classifiers Trained with Data After Discretization

Since J48 decision and SVM can handle both nominal and numerical data, we would like to train them with our data before discretization as well.

Firstly, we will use the data which we cleaned from task 1, then compare them with discretized data side by side.

While the common measures for evaluating the data mining results include:

1. Accuracy, 2. Precision, 3. Recall, 4. F-Measure, 5 ROC Curve

We want our classifiers to classify mobiles into high-price category and low-price category correctly, we want to have both highest True Positive and True Negative rates. Here, we chose accuracy to evaluate our 4 classifiers.

3.1.1 J48 Decision Tree

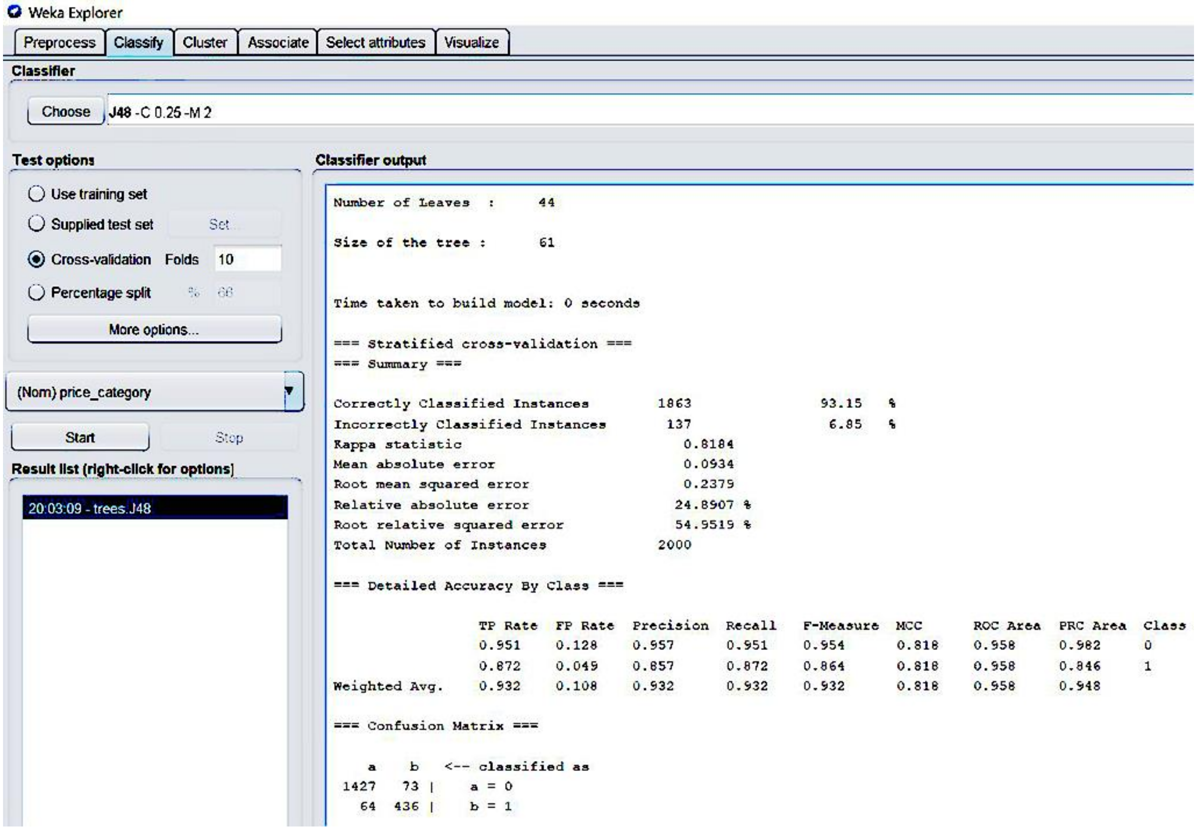


Figure 8. J48 Decision Tree with data after discretization, showing accuracy 93.15%

3.1.2 SVM

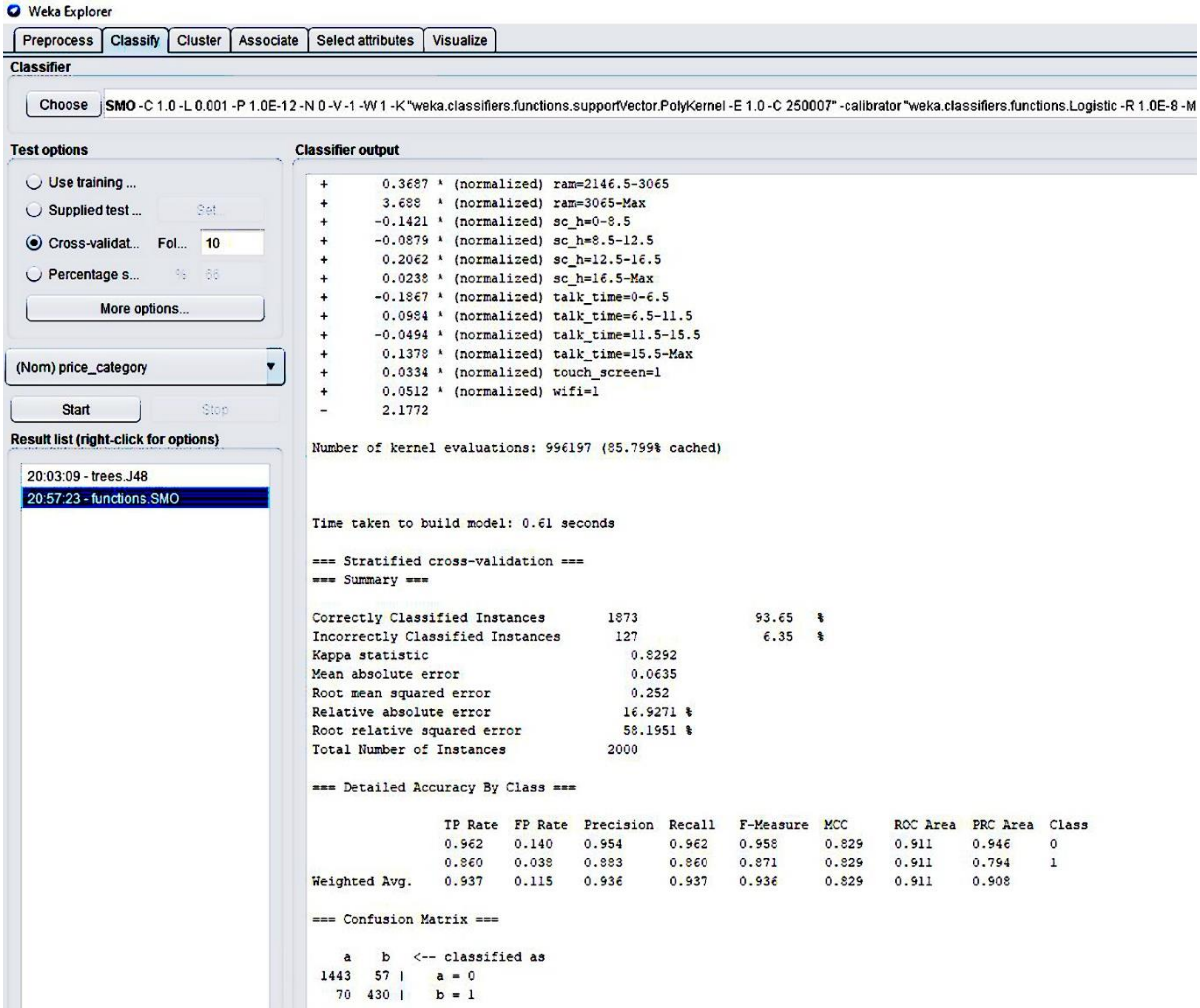


Figure 9. SVM with data after discretization, showing accuracy 93.65%

3.2 Classifiers Trained with Data Before Discretization

3.2.1 J48 Decision Tree

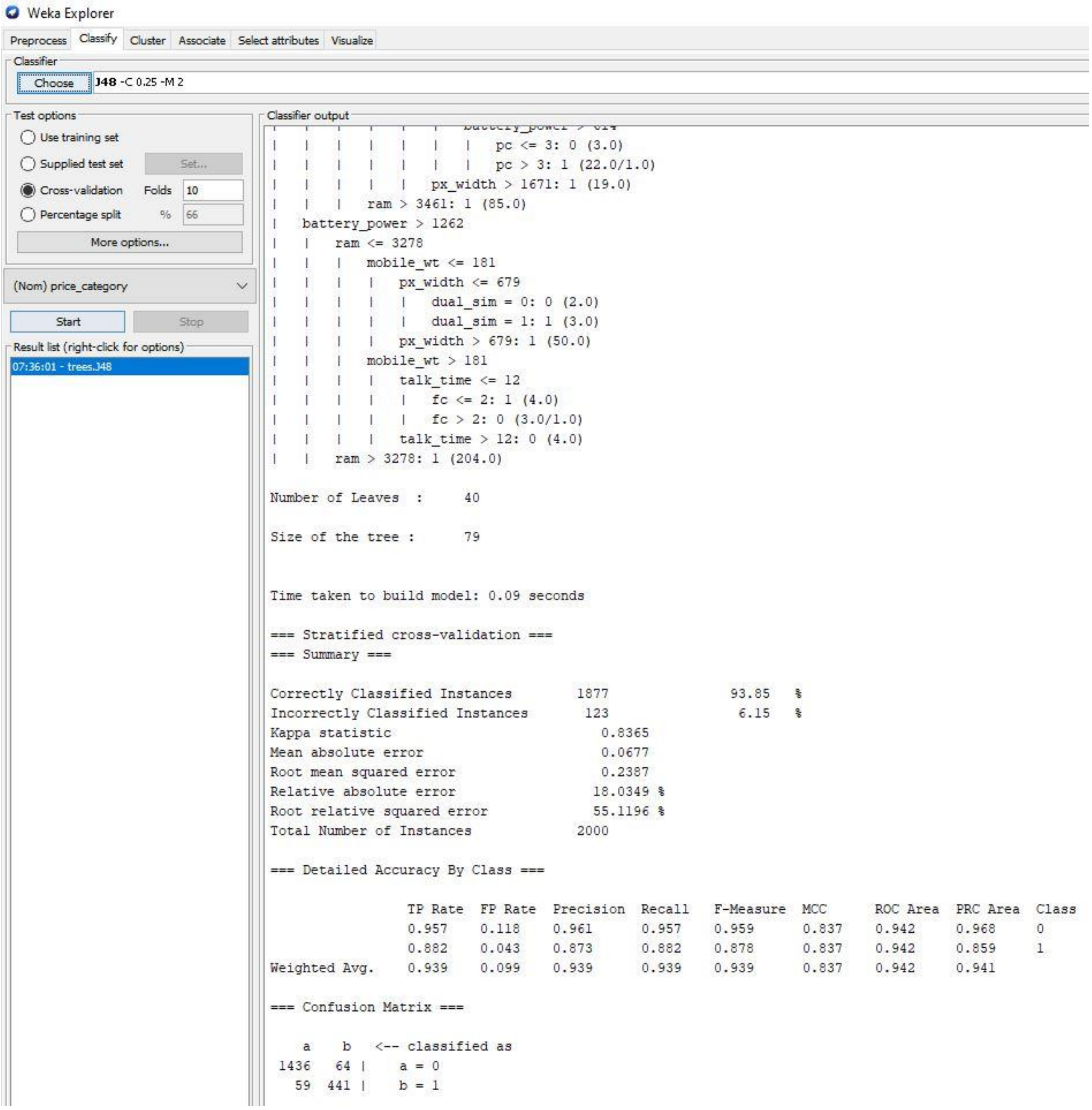


Figure 10. J48 Decision Tree with data after discretization showing accuracy 93.85%

3.2.2 SVM

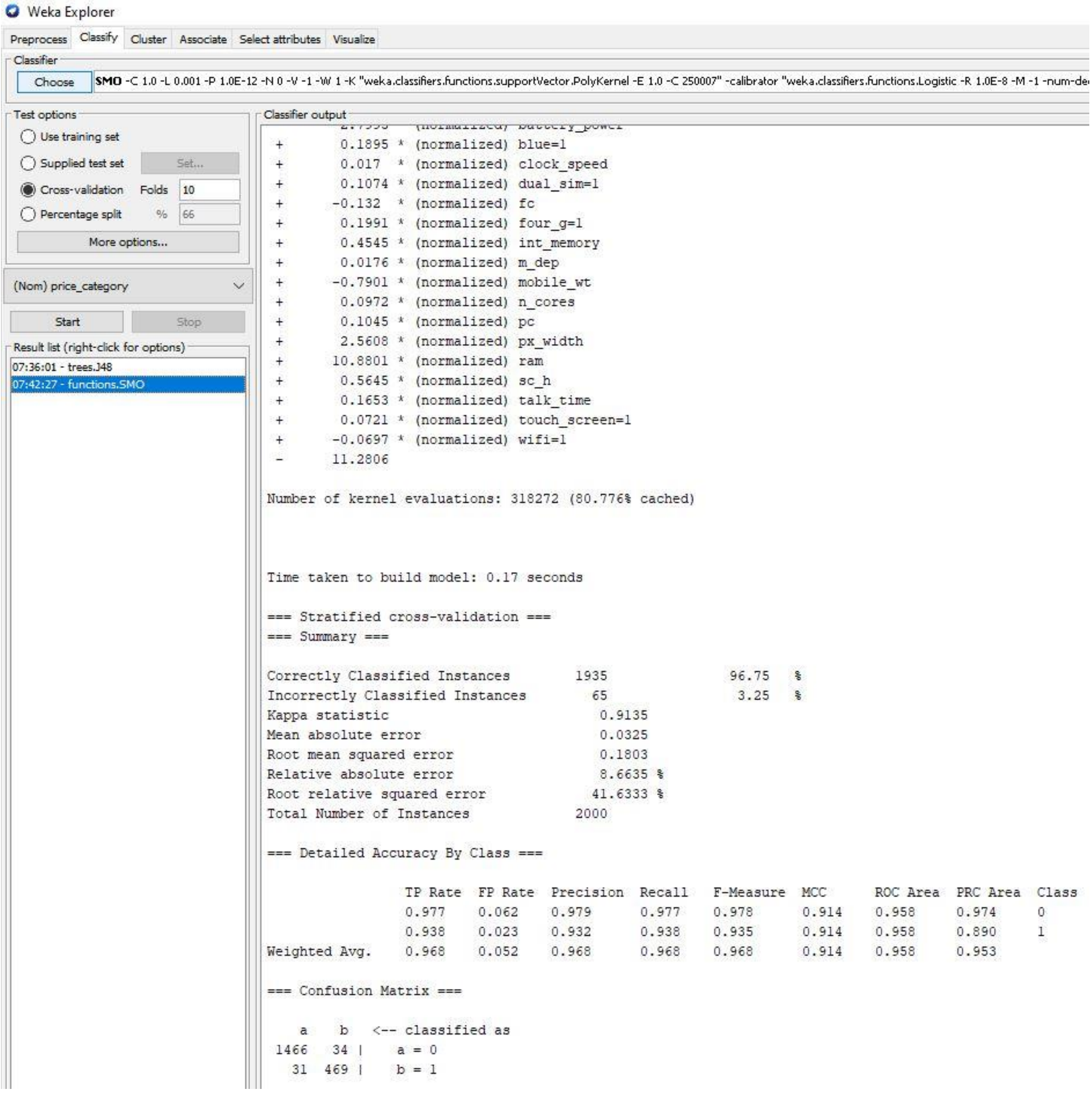


Figure 11. SVM with data after discretization, showing accuracy 96.75%

3.3 Evaluation

We used 10-Fold Cross Validation. With our data after discretization, the accuracy of SVM is slightly higher than J48 decision tree but not by much at all.

With our data before discretization, the accuracy of SVM is the highest among all our 4 classifiers.

	J48 Decision Tree	*SVM*
Data After Discretization	93.15%	93.65%
Data Before Discretization	93.85%	**96.75%**

Table 2. Accuracy Comparison of J48 Decision Tree and SVM with data before and after discretization

So far, we can observe two conclusions here:

1. SVM performs better than J48 Decision Tree with our mobile dataset.
2. It is better to keep some of the mobile attributes in numerical data type for training models with better accuracy.

4. DBSCAN CLUSTERING

There is no one clustering algorithm that suits all kinds of data. We have tried K-Means and EM with our dataset. However, most of the results have incorrectly-clustered-instance rate above 40%. In comparison, DBSCAN is performing better. Hence, we chose DBSCAN for analysing our data here.

4.1 Curse of Dimensionality

Clustering generally depends on some sort of distance measure. Points near each other are in the same cluster; points far apart are in different clusters. However, in high dimensional spaces, distance measures do not work very well.

The common theme of these problems is that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse. Hence, we reduced our dimensions first so that the distance metric will make more sense here.

4.2 Result

Our goal is to get a low incorrectly-clustered-instance rate with as few unclustered instances as possible. After several experiments, we chose “battery_power”, “ram”, “fc” and “pc” with epsilon = 0.1 and minPoints = 6 for our final model, which seemed to have a better result compared to other models. We got 24 clusters with incorrectly-clustered-instance rate 10.4%, which is not bad. DBSCAN treated 1442 instances as noise and did not cluster them (Figure 12).

```
Unclustered instances : 1442

Class attribute: price_category
Classes to Clusters:

   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  <-- assigned to cluster
 302   1   8   2   6   6  11   0   0   0   7  17   5   6   2   6   6   6   6   9   6   6   6   0 | 0
  44  48   1   5   0   0   0   6   6  10   0   0   3   0   4   0   0   0   0   0   0   0   0   7 | 1

Cluster  0 <-- 0
Cluster  1 <-- 1
Cluster  2 <-- No class
Cluster  3 <-- No class
Cluster  4 <-- No class
Cluster  5 <-- No class
Cluster  6 <-- No class
Cluster  7 <-- No class
Cluster  8 <-- No class
Cluster  9 <-- No class
Cluster 10 <-- No class
Cluster 11 <-- No class
Cluster 12 <-- No class
Cluster 13 <-- No class
Cluster 14 <-- No class
Cluster 15 <-- No class
Cluster 16 <-- No class
Cluster 17 <-- No class
Cluster 18 <-- No class
Cluster 19 <-- No class
Cluster 20 <-- No class
Cluster 21 <-- No class
Cluster 22 <-- No class
Cluster 23 <-- No class

Incorrectly clustered instances :      208.0     10.4    %
```

Figure 12. Result of DBSCAN clustering

4.3 Plots & Patterns

For better visualisation, we set Cluster as the Y axis and attributes as X axis; blue colour for price_category = 0 and red colour for price_category = 1. As our target clusters are cluster 0 and cluster 1, which are at the bottom part of the Y axis. Let us see if there are any interesting patterns:

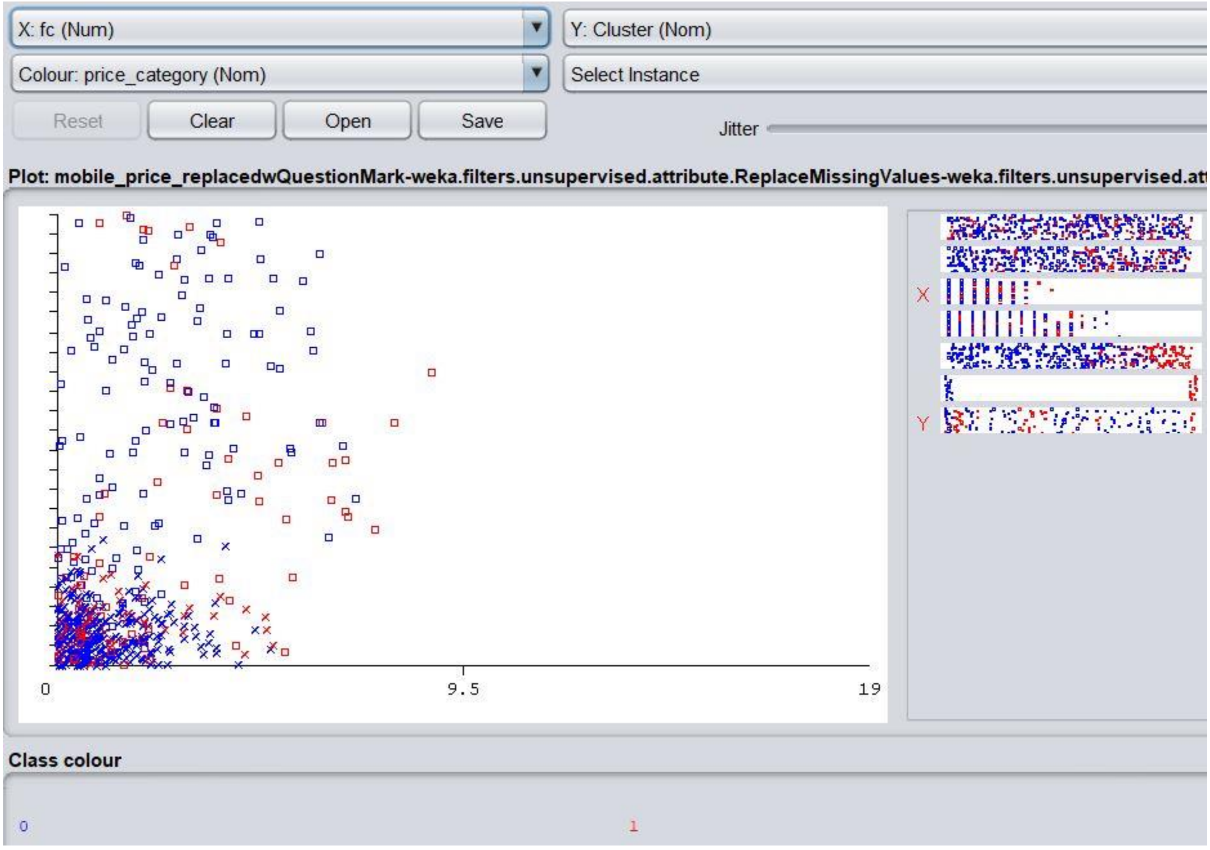


Figure 13. Plot of front camera and cluster

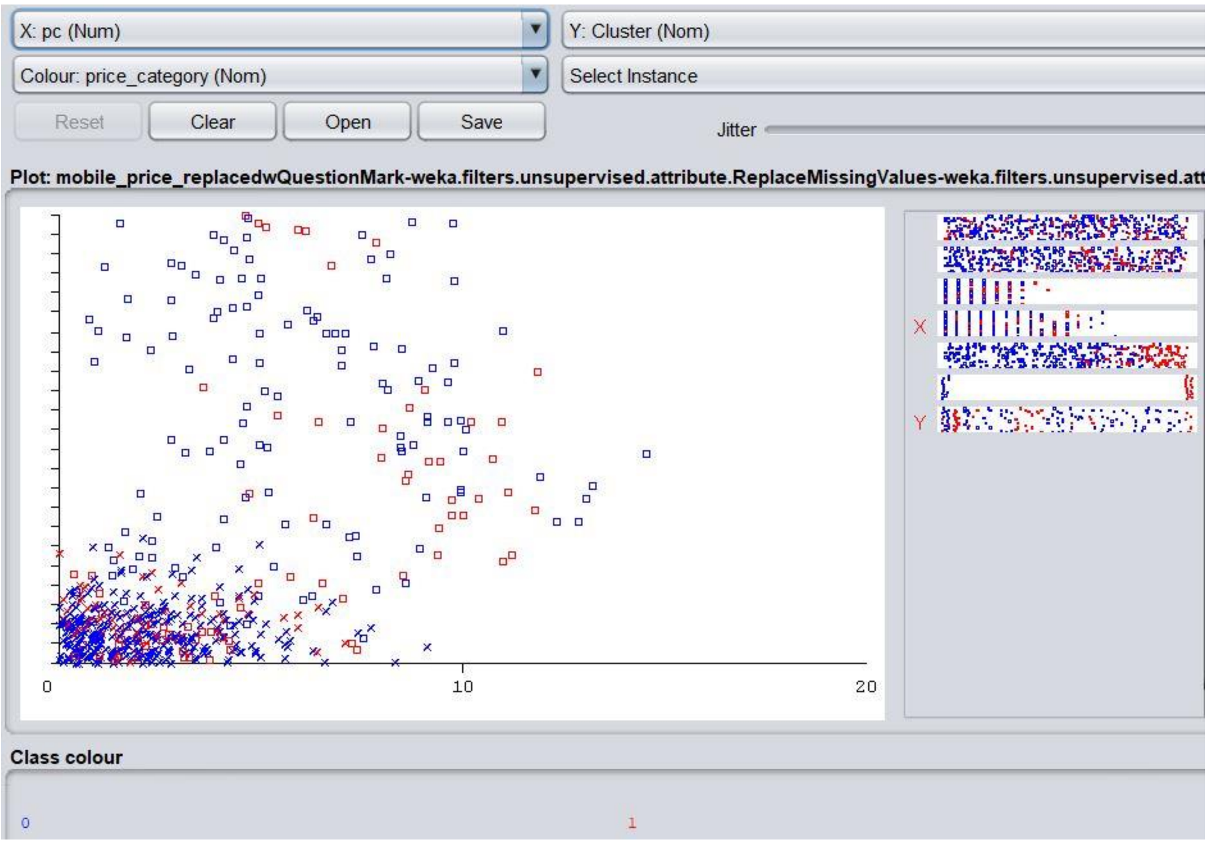


Figure 14. Plot of primary camera and cluster

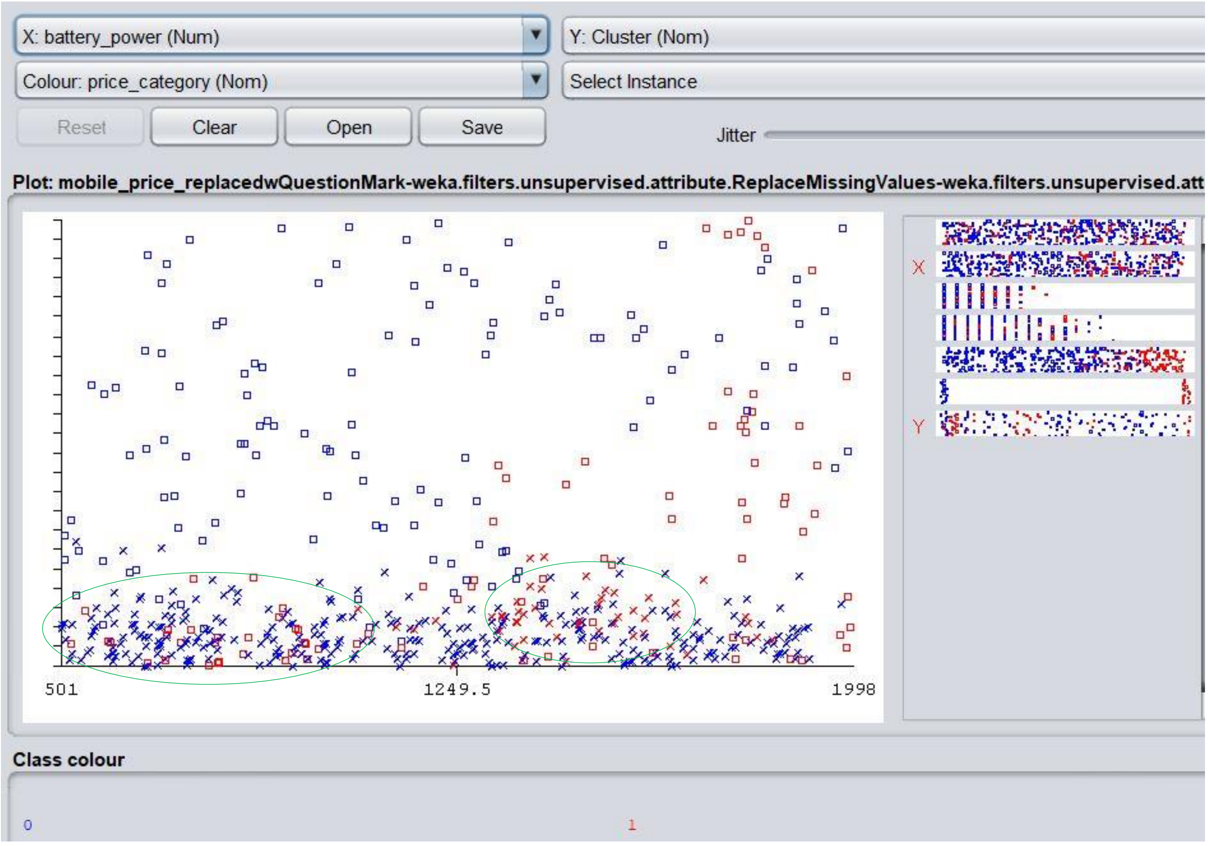


Figure 15. Plot of battery power and cluster

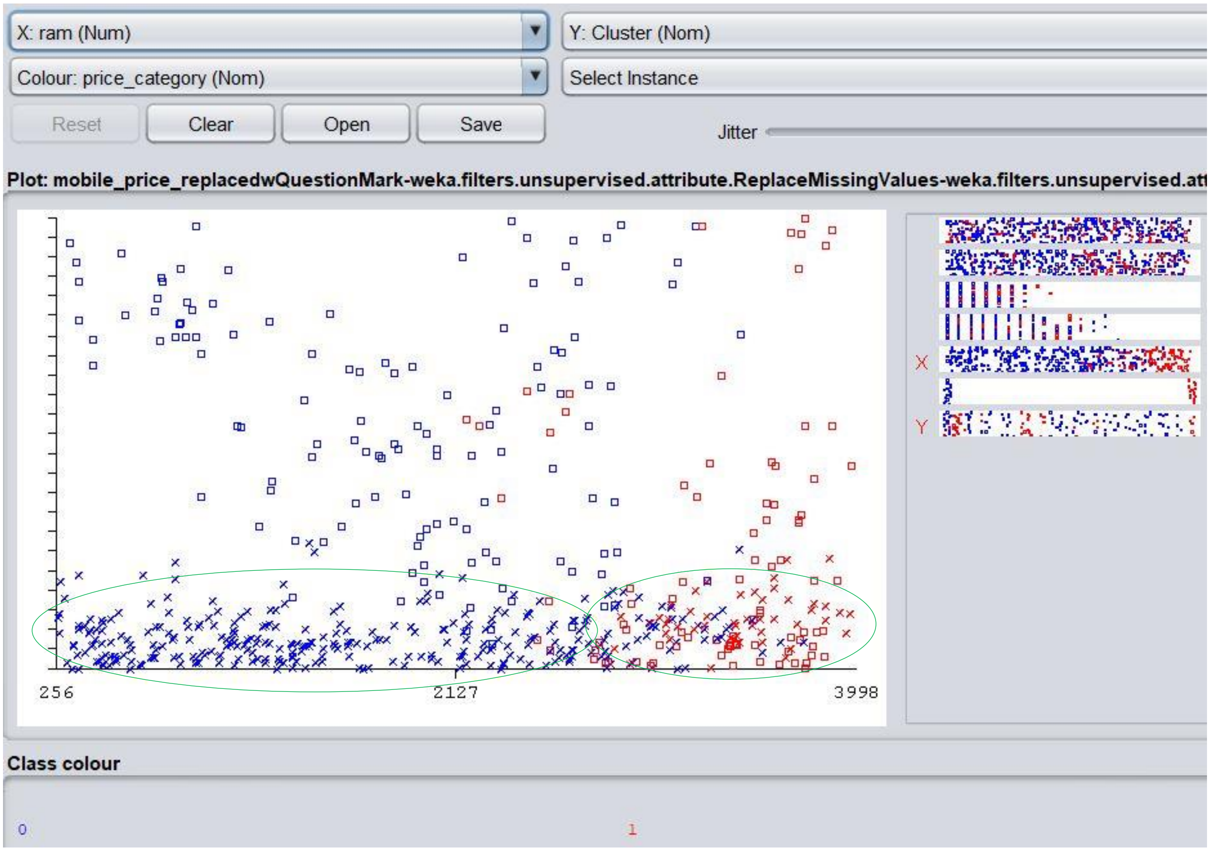


Figure 16. Plot of ram and cluster

From the plots above, we can see there are no obvious patterns with fc and pc, but a light density pattern with battery_power, and undoubtedly a strong pattern with ram.

4.4 Verdict

Once again, we have proved that there is such a strong association between “ram” and “price_category”. Most of the times, the other attributes do not play big parts in our models. They have been even treated as “noise” in building models because of their weak associations compared to “ram”.

That is also why DBSCAN performs better in this task. DBSCAN is good at dealing with noise and outliers, and performs well with clusters in arbitrary shapes.

5. DATA REDUCTION

5.1 Numerosity Reduction

5.1.1 Stratified Random Sampling

Here we imported our cleaned data into Excel to do SRS (Stratified Random Sampling)

We used function Rand() to create a column of random numbers between 0 and 1.

fx =RAND()

	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
0	1	0	7	0.6	188	2	2	756	2549	9	19	0	1	0	0.843955
1	0	1	53	0.7	136	3	6	1988	2631	17	7	1	0	0	0.990295
1	2	1	41	0.9	145	5	6	1716	2603	11	9	1	0	0	0.020711
0	0	0	10	0.8	131	6	9	1786	2769	16	11	0	0	0	0.645228
0	13	1	44	0.6	141	2	14	1212	1411	8	15	1	0	0	0.686018
1	3	0	22	0.7	164	1	7	1654	1067	17	10	0	0	0	0.580897
0	4	1	10	0.8	139	8	10	1018	3220	13	18	0	1	1	0.613685
1	0	0	24	0.8	187	4	0	1149	700	16	5	1	1	0	0.550041
0	0	0	53	0.7	174	7	14	836	1099	17	20	0	0	0	0.329769
1	2	1	9	0.1	93	5	15	1224	513	19	12	0	0	0	0.182744
1	0	0	9	0.1	182	5	1	874	3946	5	7	0	0	1	0.439698
0	5	1	33	0.5	177	8	18	1005	3826	14	13	1	1	1	0.523553
0	2	0	33	0.6	159	4	17	748	1482	18	2	0	0	0	0.680672
0	7	0	17	1	198	4	11	1440	2680	7	4	0	1	0	0.499384
0	13	1	52	0.7	185	1	17	563	373	14	3	0	1	0	0.000652
0	3	0	46	0.7	159	2	16	1864	568	17	11	1	1	0	0.304397
0	1	1	13	0.1	196	8	4	1850	3554	10	19	0	1	1	0.220565
1	7	1	23	0.1	121	3	17	810	3752	10	18	1	0	1	0.574002
1	11	0	49	0.6	101	5	18	878	1835	19	16	1	0	0	0.490655
0	4	0	19	1	121	4	11	1064	2337	11	18	1	1	0	0.703253
1	12	0	39	0.8	81	7	14	1854	2819	17	3	1	0	1	0.402794
0	1	0	13	1	156	2	2	1385	3283	17	15	0	0	1	0.341192

Figure 17. Generate random numbers for each instance in Excel

Then we used SORT to sort out the 2000 instances first by price_category, second by the random numbers we created.

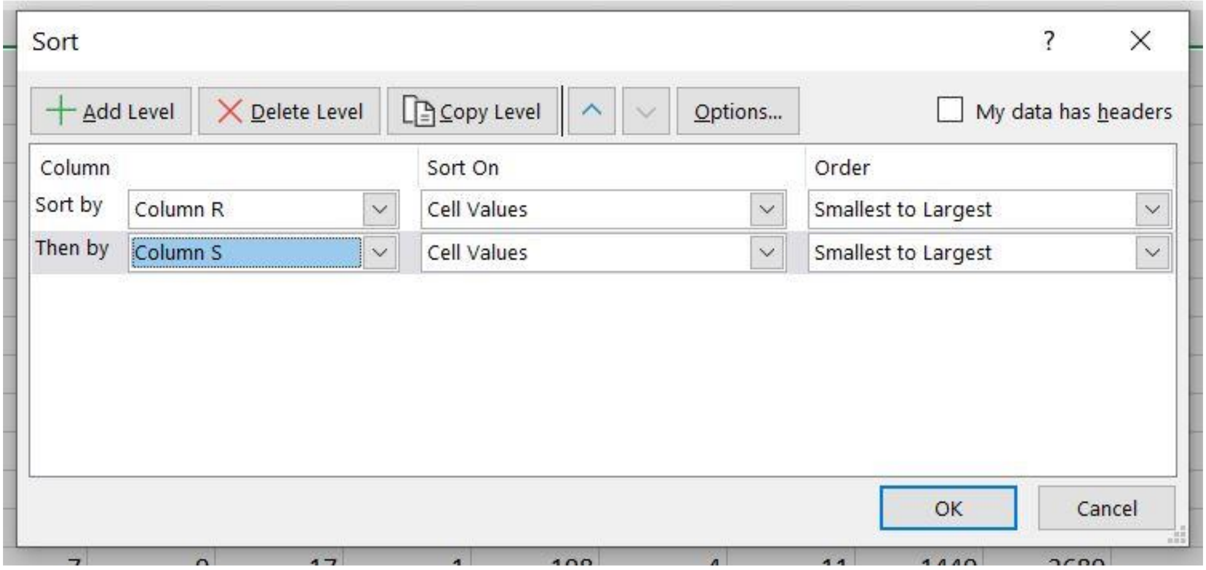
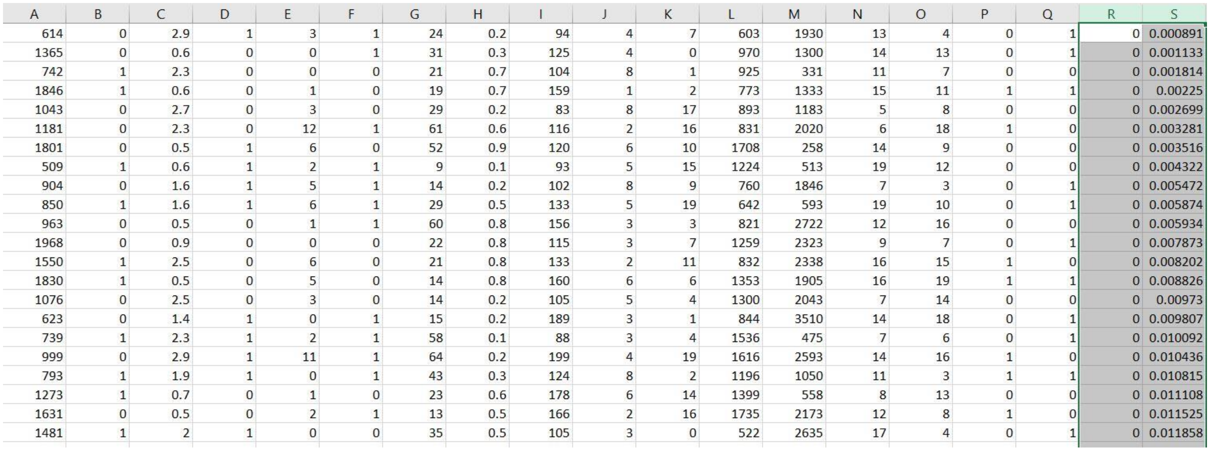


Figure 18. Sort instances in Excel

Then we got 2 strata of data (price_category = 0 and price_category = 1) randomly distributed:



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
614	0	2.9	1	3	1	24	0.2	94	4	7	603	1930	13	4	0	1	0	0.000891
1365	0	0.6	0	0	1	31	0.3	125	4	0	970	1300	14	13	0	1	0	0.001133
742	1	2.3	0	0	0	21	0.7	104	8	1	925	331	11	7	0	0	0	0.001814
1846	1	0.6	0	1	0	19	0.7	159	1	2	773	1333	15	11	1	1	0	0.00225
1043	0	2.7	0	3	0	29	0.2	83	8	17	893	1183	5	8	0	0	0	0.002699
1181	0	2.3	0	12	1	61	0.6	116	2	16	831	2020	6	18	1	0	0	0.003281
1801	0	0.5	1	6	0	52	0.9	120	6	10	1708	258	14	9	0	0	0	0.003516
509	1	0.6	1	2	1	9	0.1	93	5	15	1224	513	19	12	0	0	0	0.004322
904	0	1.6	1	5	1	14	0.2	102	8	9	760	1846	7	3	0	1	0	0.005472
850	1	1.6	1	6	1	29	0.5	133	5	19	642	593	19	10	0	1	0	0.005874
963	0	0.5	0	1	1	60	0.8	156	3	3	821	2722	12	16	0	0	0	0.005934
1968	0	0.9	0	0	0	22	0.8	115	3	7	1259	2323	9	7	0	1	0	0.007873
1550	1	2.5	0	6	0	21	0.8	133	2	11	832	2338	16	15	1	0	0	0.008202
1830	1	0.5	0	5	0	14	0.8	160	6	6	1353	1905	16	19	1	1	0	0.008826
1076	0	2.5	0	3	0	14	0.2	105	5	4	1300	2043	7	14	0	0	0	0.00973
623	0	1.4	1	0	1	15	0.2	189	3	1	844	3510	14	18	0	1	0	0.009807
739	1	2.3	1	2	1	58	0.1	88	3	4	1536	475	7	6	0	1	0	0.010092
999	0	2.9	1	11	1	64	0.2	199	4	19	1616	2593	14	16	1	0	0	0.010436
793	1	1.9	1	0	1	43	0.3	124	8	2	1196	1050	11	3	1	1	0	0.010815
1273	1	0.7	0	1	0	23	0.6	178	6	14	1399	558	8	13	0	0	0	0.011108
1631	0	0.5	0	2	1	13	0.5	166	2	16	1735	2173	12	8	1	0	0	0.011525
1481	1	2	1	0	0	35	0.5	105	3	0	522	2635	17	4	0	1	0	0.011858

Figure 19. Instances sorted result in Excel

Then we chose some certain percent of data from price_category = 0, and some certain percent of data from price_category = 1.

5.1.2 J48 Decision Tree & SVM

Here, we repeated the task 3 all over again (please refer to step 3 for details) with the top 10%, 30%, 50% and 70% stratified random samples, then compared the differences.

Stratified Random Samples	*J48 Decision Tree*	*SVM*
Top 10% Data	92.50%	95.50%
Top 30% Data	91.83%	96.00%
Top 50% Data	93.10%	96.10%

Top 70% Data	**93.93%**	**96.43%**
Entire Data	93.85%	96.75%

Table 3. Accuracy comparison between J48 Decision Tree and SVM with different stratified random samples

5.1.3 Verdict

Theoretically speaking, in big data, numerosity reduction can help us save a lot of time and cost on analysing the entire data by replacing the original dataset with a sparse representation of the data.

In our case, the accuracy tends to decrease while we reduce our data size (Figure 20), as our dataset is not big, which has only 2,000 instances. If we must do numerosity reduction, reducing our data by 30% could be a good option, since the accuracy of J48 increases a little bit and the accuracy of SVM only drops a little bit.

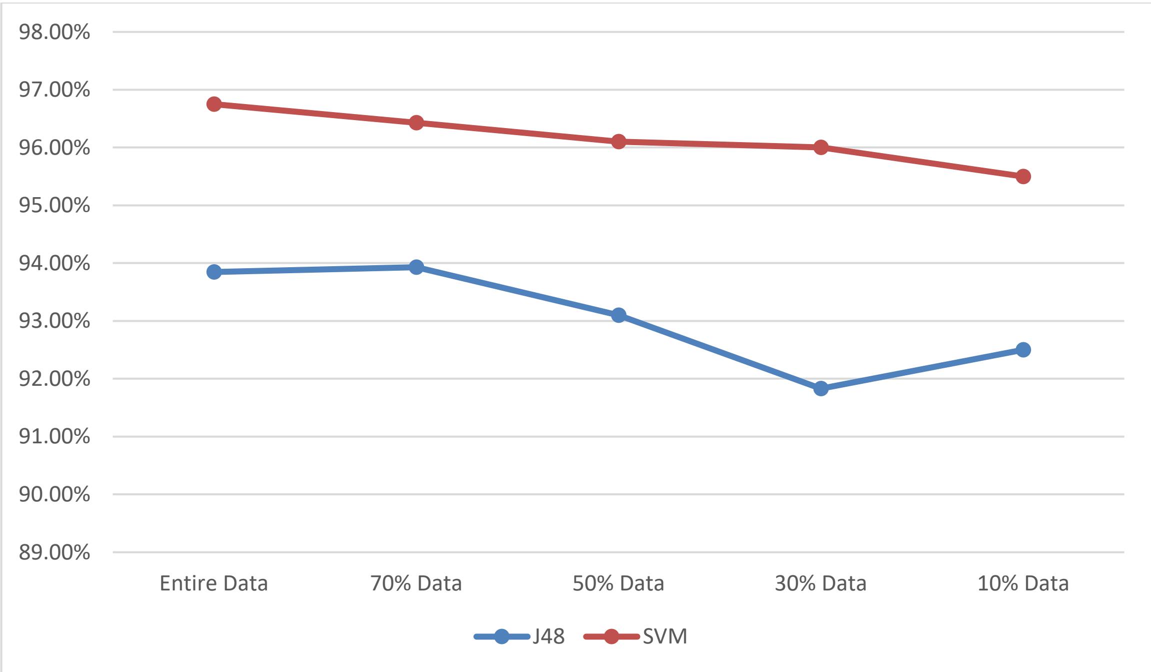


Figure 20. Accuracy chart of J48 Decision Tree and SVM with different data size

5.2 Attribute Reduction

We use Gain Ratio Attribute Eval to select the best ranked attributes. Here we chose the top 8 ranked attributes and remove the rest to train our models.

5.2.1 Classifiers Trained with Data After Discretization

```
Ranked attributes:
 0.243521    13 ram
 0.0118511    1 battery_power
 0.009479    12 px_width
 0.0019101    7 int_memory
 0.0017212   14 sc_h
 0.0016451    9 mobile_wt
 0.0007849    6 four_g
 0.0005665   11 pc
 0.0005263   10 n_cores
 0.0005089    2 blue
 0.0004047    4 dual_sim
 0.0003603    5 fc
 0.0003297   15 talk_time
 0.0002937    3 clock_speed
 0.0002782   17 wifi
 0.0002171    8 m_dep
 0.0000471   16 touch_screen

Selected attributes: 13,1,12,7,14,9,6,11,10,2,4,5,15,3,17,8,16 : 17
```

Figure 21. Attributes ranked by Gain Ratio Attribute Eval

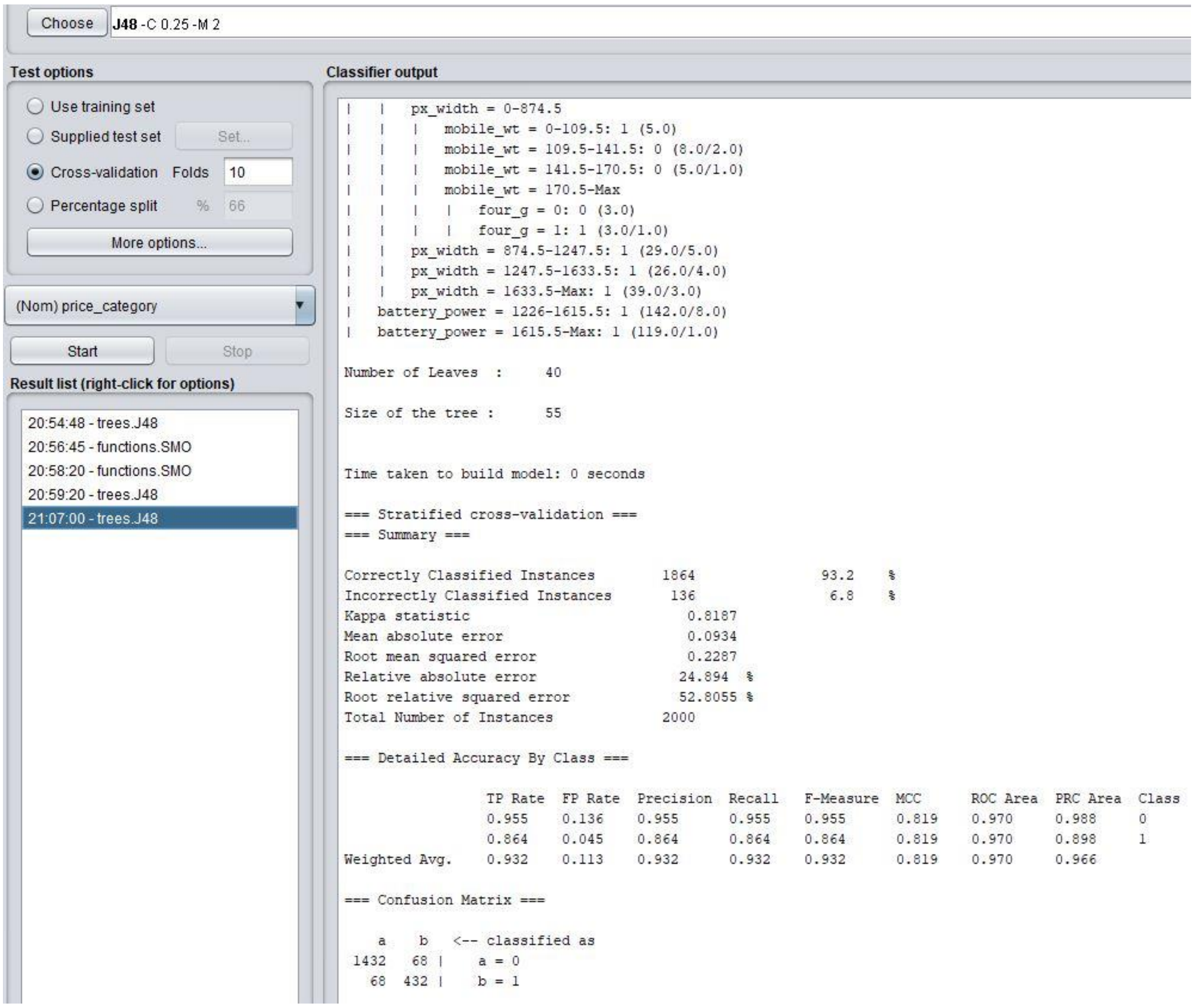


Figure 22. Result of J48 Decision Tree with selected attributes after discretization

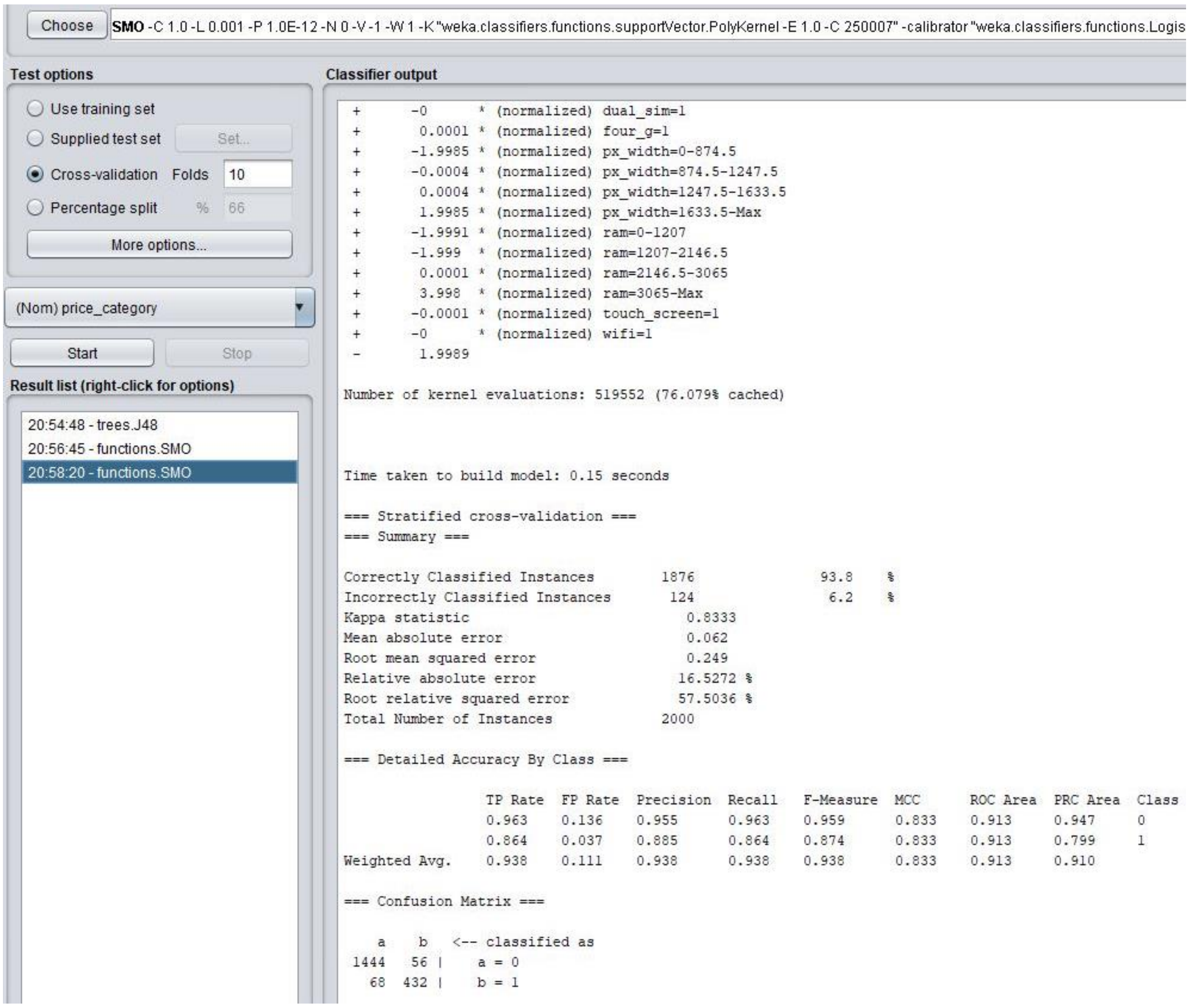


Figure 23. Result of SMO with selected attributes after discretization

5.2.2 Classifiers Trained with Data Before Discretization

Reprocess the steps in 5.2.1 with data before discretization

```
Ranked attributes:
 0.2804926  13 ram
 0.0261579   1 battery_power
 0.0211672  12 px_width
 0.0007849   6 four_g
 0.0005089   2 blue
 0.0004047   4 dual_sim
 0.0002782  17 wifi
 0.0000471  16 touch_screen
 0          10 n_cores
 0           3 clock_speed
 0          15 talk_time
 0          14 sc_h
 0           5 fc
 0          11 pc
 0           7 int_memory
 0           8 m_dep
 0           9 mobile_wt

Selected attributes: 13,1,12,6,2,4,17,16,10,3,15,14,5,11,7,8,9 : 17
```

Figure 24. Attributes ranked by Gain Ratio Attribute Eval

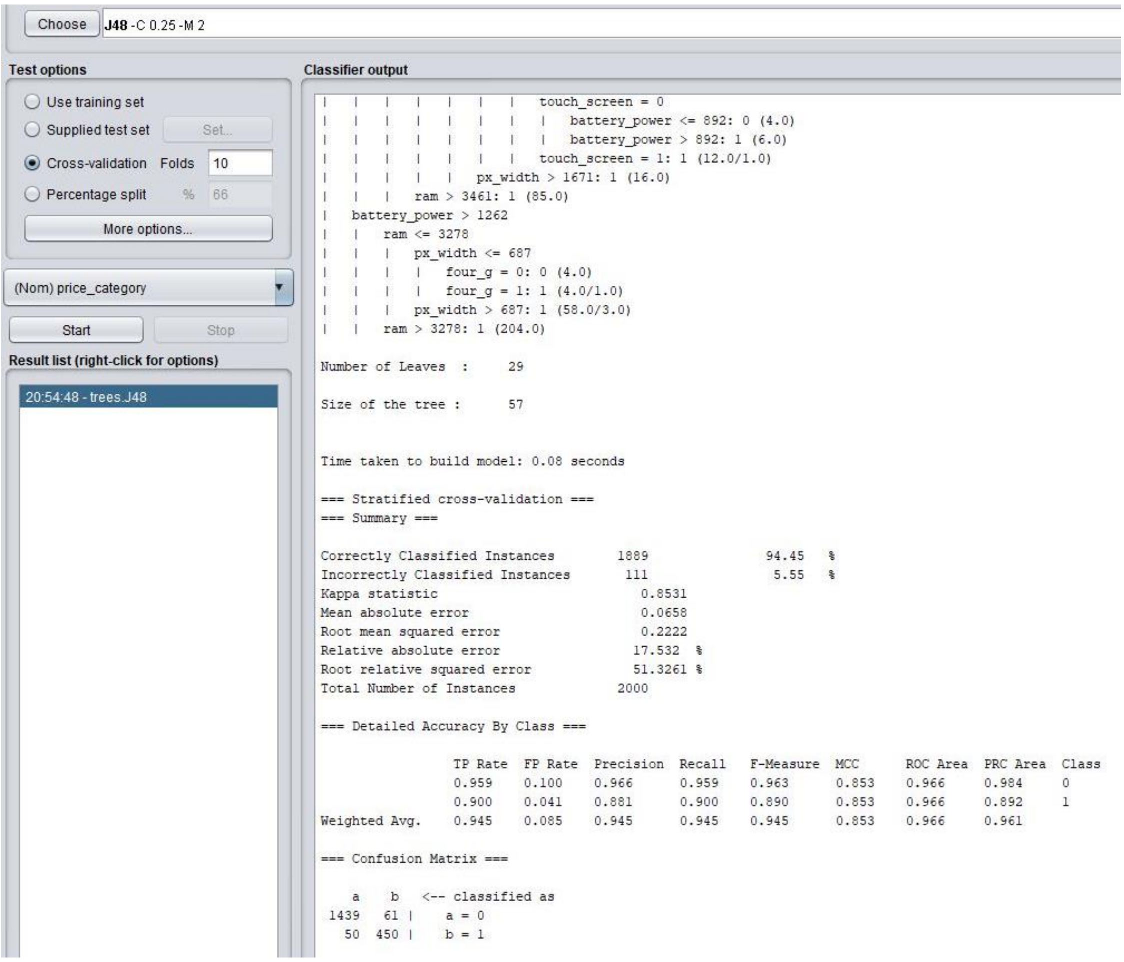


Figure 25. Result of J48 Decision Tree with selected attributes after discretization

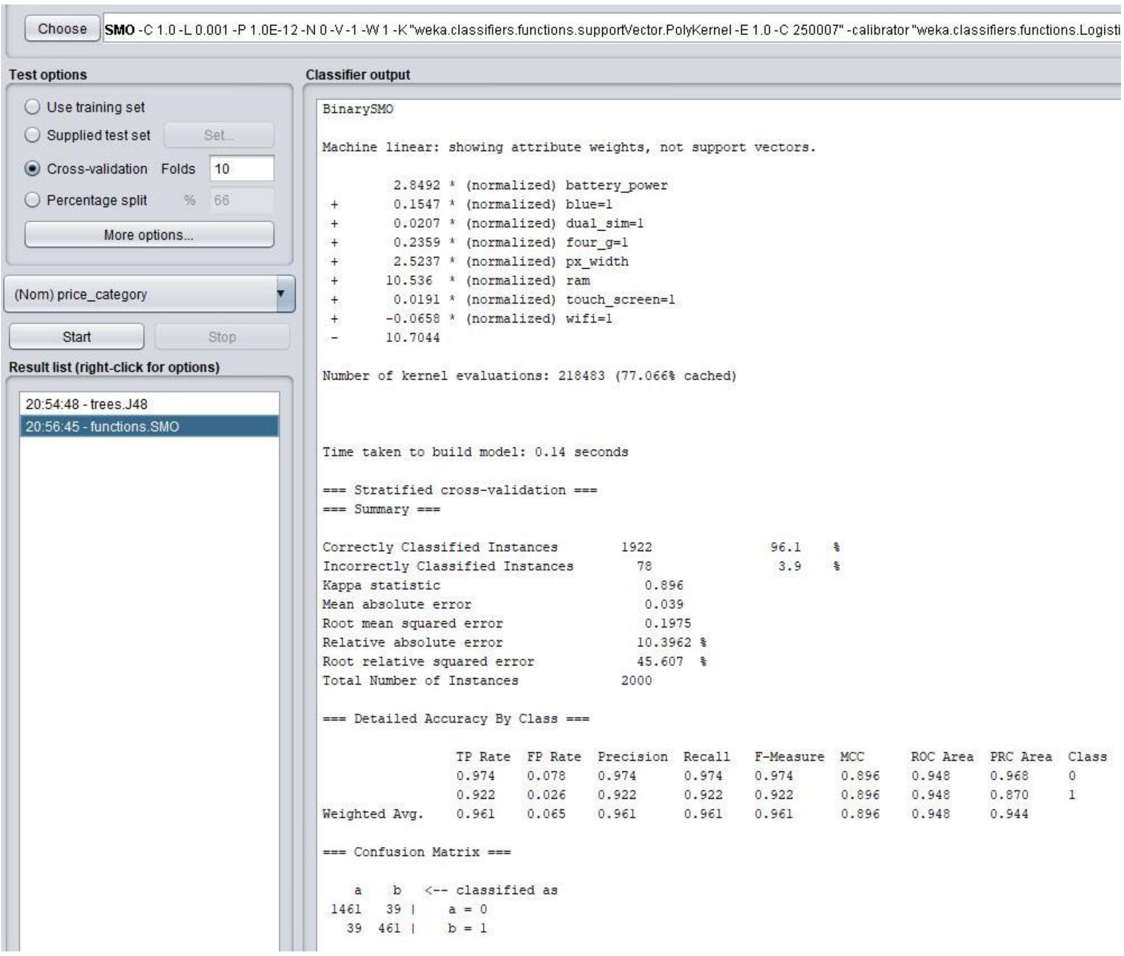


Figure 26. Result of SMO with selected attributes before discretization

5.2.3 Verdict

From the comparison of the first two tables below, we can see the accuracy of 3 out of 4 classifiers have been improved, but the accuracy of our best performed classifier dropped.

Also, even if we removed all the attributes and left only the top ranked attribute “ram”, the accuracy of our 4 models was still all above 91% (Table 6). If we removed “ram” but keep all the other attributes, the accuracy of all our models dropped significantly (Table 7). It proves that there is such a strong association between “ram” and “price_category”.

After Data Cleaning	*J48 Decision Tree*	*SVM*
Data After Discretization	93.15%	93.65%
Data Before Discretization	93.85%	96.75%

Table 4. Accuracy comparison of J48 Decision Tree and SVM with data after and before discretization after cleaning

After Attribute Reduction	*J48 Decision Tree*	*SVM*
Data After Discretization	**93.2%**	**93.8%**
Data Before Discretization	**94.45%**	96.1%

Table 5. Accuracy comparison of J48 Decision Tree and SVM with data after and before discretization after reduction

Only 1 Attribute “ram”	*J48 Decision Tree*	*SVM*
Data After Discretization	91.7%	91.7%
Data Before Discretization	92.15%	91.7%

Table 6. Accuracy comparison of J48 Decision Tree and SVM with data after and before discretization with one attribute ram

Remove “ram” Only	*J48 Decision Tree*	*SVM*

Data After Discretization	74.8%	75%
Data Before Discretization	66.65%	75%

Table 7. Accuracy comparison of J48 Decision Tree and SVM with data after and before discretization with all attributes beside ram

Theoretically speaking, attribute selection or attribute reduction is for decreasing the complexity and improving the accuracy. However, this is not always necessarily the truth. While attribute selection / reduction does decrease complexity, it does not have to improve accuracy.

In our case, due to the strong association between “ram” and “price_category” and such low-ranking scores of other attributes shown in both of the Gain Ratio Analysis figure above(figure 21 and figure 24), no matter how we train our classifiers, there will not be much difference (within 6% approximately), as long as the attribute “ram” is selected.

6. COMPARISON OF 3 MINING METHODS

6.1 Association Rule Mining

Association Rule Mining is a rule-based machine learning method to discover interesting relations between variables. The Apriori Algorithm we used here is usually an unsupervised learning, but it can still be used for labelled data. From our test result, we can see Apriori Algorithm mine out 3 interesting association rules for us, and all of them contain our dominant variable “ram”. We will explain the pros and cons of this method for handling our mobile data, and the comparison with other two methods below:

6.1.1 Pros

A	B
FOUR_G=1 RAM='(3065-INF)' 271	price_category=1 232
RAM='(3065-INF)' 500	price_category=1 417
DUAL_SIM=1 RAM='(3065-INF)' 278	price_category=1 230

Table 8. Association rule mining result

- Association rule mining makes sure that the percentage of A and B happening is significant by setting the minimum support parameter.
- The number behind the attributes tells us the likelihood between A and B. For example: ram='(3065-inf)' 500 ==> price_category=1 417
- It shows us multiple features in A, which can provide mobile companies more information on their mobile designing process.

6.1.2 Cons

- Because the data distribution of price_category = 0 and price_category = 1 is quite unbalanced in our dataset, where the ratio is 1500 : 500, when we set our minimum support too high, it is impossible to find the associations we want.

6.1.3 Comparison

- Association rule mining requires nominal variables; therefore, we had to discretize our data first. It could lose some important information after discretization, and how to choose the proper bin number for discretization can be a challenge too.

- The resulting rules are more intuitive and easier to interpret or communicate to mobile companies.

6.2 Classification

Classification Data Mining is a process of finding a model that describes and distinguishes data classes and concepts. Here, we used J48 Decision Tree and SVM. They both have very high accuracy, while "ram" is one of the selected attributes (see task 5.2.3 above for more details), for example, "ram" is always at the top of our decision trees. We will explain the pros and cons of this method for handling our mobile data, and the comparison with other two methods below:

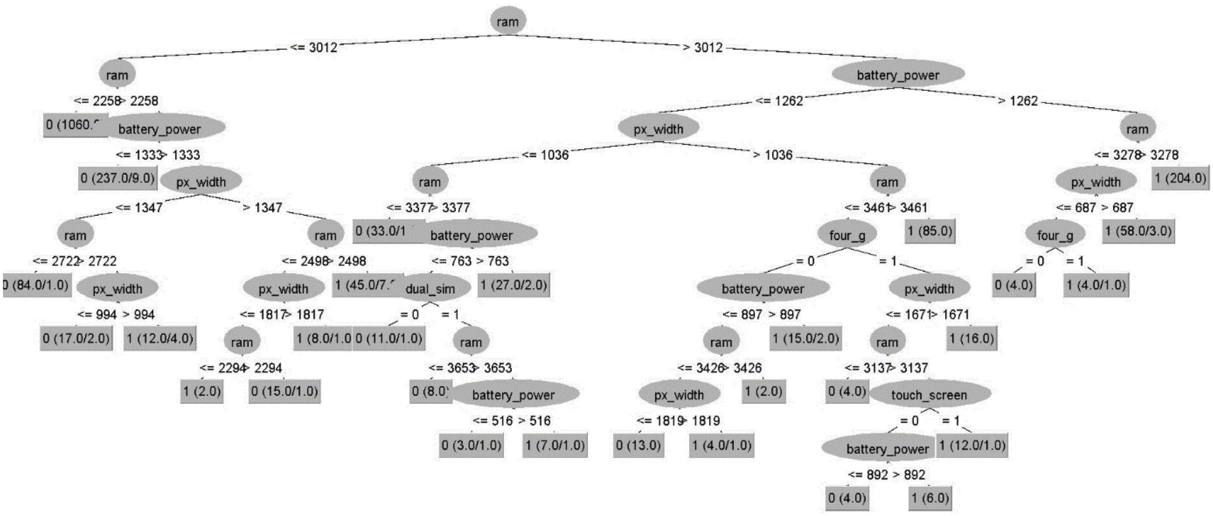


Figure 27. J48 Decision Tree visualization

6.2.1 Pros

- Decision Trees are perfect for visual presentation to mobile companies.

- SVM performs relatively well in high dimensional space.

6.2.2 Cons

- Both J48 and SVM can be affected by noise quite easily. However, it did not really happen in our case as we have a dominant attribute "ram".

- SVM models are difficult to be interpreted or visualised.

6.2.3 Comparison

- Not like Association Rule Mining that can only work with categorical data or clustering that generally performs better on numerical data, both J48 and SVM can work with numerical and categorical data, so it requires less data pre-processing.

- Both J48 and SVM are supervised learning, and both work well with our labelled data, so if we must choose one among these 3 methods, classification mining would be our first choice.

6.3 Clustering

Clustering is a way to group a set of data points in a way that similar data points are grouped together. Clustering is an unsupervised learning method, so there is usually no label associated with data points. However, in this project we still gave it a go with our mobile dataset.

In most of our experimental models, there was hardly any patterns to be seen in the cluster distribution plots. However, through our final model, we can still see an obvious pattern with the dominant attribute "ram". We will explain the pros and cons of this method compared to the other two methods below:

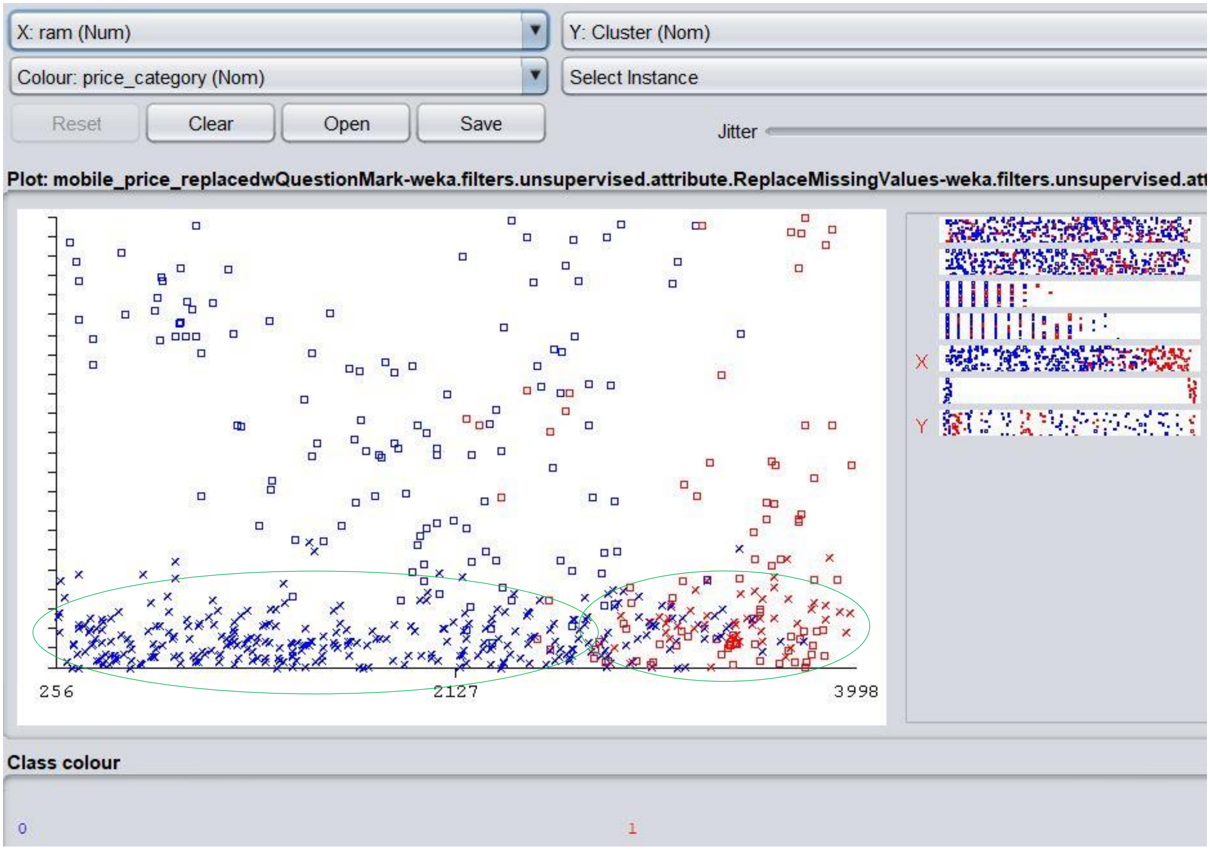


Figure 28. The plot of ram and price cluster

6.3.1 Pros

- We do not need to specify the number of clusters for DBSCAN, which is a great advantage of DBSCAN over K-means.

- DBSCAN is robust to noise and outliers, whereas K-mean struggles with outliers.

- DBSCAN performs well with clusters in arbitrary shapes, which K-mean would not be able to find.

6.3.2 Cons

- Most of the clustering algorithms perform the best on continuous data, so whether and how we pre-process the data from nominal to numeric or binary is always a challenge, as we do not know what the best procedure is to get the best results. Thus, a lot of experimental steps are usually involved.

- Determining an appropriate eps and minPoints can be challenging.

6.3.3 Comparison

- Training set is not required to produce clusters, whereas training set is required for classification learning.

- Clustering is great at dealing with unlabelled data. For data like our mobile dataset, we would not choose clustering methods but classification or association rule mining.