THE UNIVERSITY OF WESTERN AUSTRALIA

Achieve International Excellence

CITS5504

Data Warehousing SEM-1 2021

Project 2

Pattern Discovery and Building Predictive Models

Hang Qi 22449507

Chang Su 22993116

Division of Work

Section 1

Qi Hang works on introduction, background searching.

Section 2

Chang Su and Hang Qi read data files and description individually and discussed data through Zoom's online meeting.

Hang Qi cleans up the data through Python.

Chang Su cleans the data through MS Excel.

Do the cross check by two ETL methods.

Section 3

Hang Qi works on the methodology part, result check and content proofreading.

Chang Su responsible for implementation and analysing parts.

Section 4

Hang Qi works on the methodology part, result check and content proofreading.

Chang Su responsible for implementation in weka and analysing parts.

Section 5

Chang Su works on the methodology part, result check and content proofreading.

Hang Qi responsible for implement in Weka and analyses parts.

Section 6

Chang Su works on the methodology part, result check and content proofreading.

Hang Qi responsible for implement in Weka and analyses parts.

Section 7

Chang Su and Hang Qi do this section together with the relevant tasks in previse sections.

In the final stage, Chang Su and Hang Qi review and revise the report's contents together.

Change Su is responsible for formatting and proofreading for the early-stage vision of the report, and Hang Qi is working on final vision formatting and proofreading of the report and submitting projects.

Content

Figure Content

1 Introduction

Nowadays, mobile phones have become a necessary item for everyone. The price of a mobile phone is related to the properties of it. This project aims to produce clean, reduced or transformed data for pattern discovery and predictive analysis, using the mobile price classification dataset as the source of data. This project aims to predict whether the price of a mobile phone is high or not [1].

According to the task list, this report has 7 sections. The performance of Association Rule Mining, Classifier and Clusters in the mobile price data set is introduced. The analysis is also described in detail.

Section 2 introduces the preliminary cleaning of data sets. Section 3, 4, and 5 introduce three types of learning, including theoretical overview, WEKA experimental performance, and model analysis. Section 6 data reduction introduces numerosity reduction and attribute reduction, and the performance of different classifiers on PCA data set. Section 7 introduces the comparison of association rule mining, classification, and clustering, and the relations among three learning methods in the context of mobile data set.

* Intermediate and final result files for all data processing procedures are saved in the corresponding directories.

*Tools/Software used in the project:

- WEKA 3.8.5,
- Python 3.8.6; Python libraries: pandas 0.25.1
- Excel

2 Task 1 - Data cleaning process

The mobile_price.csv file contains a total of 22 attributes. After observing data, we found that some columns are messed. For example, dual_sim has different values that present the same meaning, Yes, YES, has, Has, etc. So are blue, three_g, and wifi. We need to convert these boolean values into binary form, which means use 0 and 1 to substitute Yes and No.

```
In [8]:
mobiledata = pd.read_csv("mobile_price.csv")

In [9]:
COLUMN_MOD = {"yes":1,"Yes":1,"has":1,"YES":1,"Has":1,"no":0,"not":0,"NO":0,"No":0,"Not":0}
COLUMN_MOD2 = {"yes":"Yes","has":"Yes","YES":"Yes","Has":"Yes","no":"No","not":"No","NO":"No","Not":"No"}

In [10]:
mobiledata.head(5)

...

In [11]:
mobiledata.loc[:,'blue'] = mobiledata.loc[:,'wifi'].replace(COLUMN_MOD)
mobiledata.loc[:,'dual_sim'] = mobiledata.loc[:,'wifi'].replace(COLUMN_MOD)
mobiledata.loc[:,'three_g'] = mobiledata.loc[:,'wifi'].replace(COLUMN_MOD)
mobiledata.loc[:,'wifi'] = mobiledata.loc[:,'wifi'].replace(COLUMN_MOD)

In [12]:
mobiledata.head(5)

Out[12]:
```

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g
0	0	842	1	2.2	1	1	0	7	0.6	188	...	20	756	2549	9	7	19	1
1	1	1021	0	0.5	0	0	1	53	0.7	136	...	905	1988	2631	17	3	7	0
2	2	563	0	0.5	0	2	1	41	0.9	145	...	1263	1716	2603	11	2	9	0
3	3	615	0	2.5	0	0	0	10	0.8	131	...	1216	1786	2769	16	8	11	0
4	4	1821	0	1.2	0	13	1	44	0.6	141	...	1208	1212	1411	8	2	15	0

5 rows × 22 columns

```
In [13]:
mobiledata.to_csv("mobile_price_01.csv")
```

Figure 1 Unify the mess data to 0 or 1

Because the project target is to raise a high-priced mobile phone plan, the first column id is not relevant. Then we remove the id column in Weka.

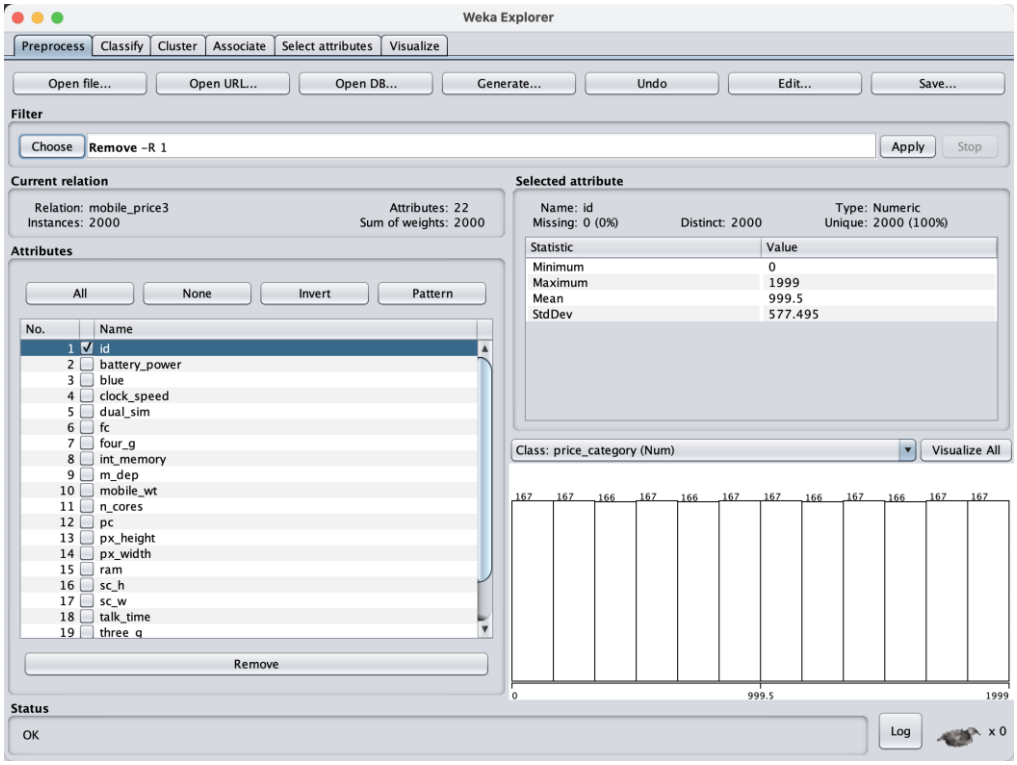


Figure 2 Remove id column

Weka shows there is no null value in the dataset, so that no null substitution is required.

Now, the data sheet is primarily cleaned. Battery_power, Clock_speed, fc, int_memory, m_dep, mobile_wt, n_cores, pc, px_height, px_width, ram, sc_h, sc_w, and talk_time are numerical. There is no need to make changes in the primary data cleanup steps. Further operation of data cleaning will be introduced in corresponding sections.

3 Tase 2 - Association rule mining

Association rule mining is a rule-based machine learning method used to discover relationships between variables in large databases. Its purpose is to identify the strong rules found in the database by the measure of interestingness [2].

3.1 Methodology

Some basic concepts are introduced here.

For all the rules $X \Rightarrow Y$:

Support

Support indicated how frequently an itemset appears in the data set. An itemset is a set of one or more items in the data set. K-itemset $X = \{x_1, x_2, \ldots, x_k\}$ Support is the proportion of transactions in the dataset which contains the itemset[3].

$$support(X \Rightarrow Y) = P(X \cup Y)$$

Confidence

Confidence is an indication of how often the rule has been found to be true. It is a conditional probability that a transaction having X also contains Y.

$$confidence(X \Rightarrow Y) = P(Y|X) = \frac{support(X \cup Y)}{support(X)}$$

Lift

Lift evaluates the degree to which the occurrence of one "lifts" the occurrence of the other, which is defined as:

$$lift(A, B) = \frac{P(A \cup B)}{P(A)P(B)}$$

If lift<1, then the occurrence of A is negatively correlated with B;

If lift>1, then the occurrence of A is positively correlated with B;

If list=1, then the occurrence of A is independent of B [4].

3.2 Implementation

First, we use the Isofrequency method to discretize the numerical attribute data in the data set. Click the choose button, select the filter / unsupervised / Discretize and click ok. Then open the Discretize filter dialog box, enter in attributeIndices corresponding attribute number (1,3,5,7,8,9,10,11,12,13,14,15,16,17). Change the bin number to 3 stands for low frequency, medium frequency and high frequency. Switch useEqualFrequency option from False to True, then click ok.

Figure 3 equal frequency discretize dialogue

Renaming the discrete data interval improves the readability of the results. The ARFF file after discrete completion is shown in the figure below.

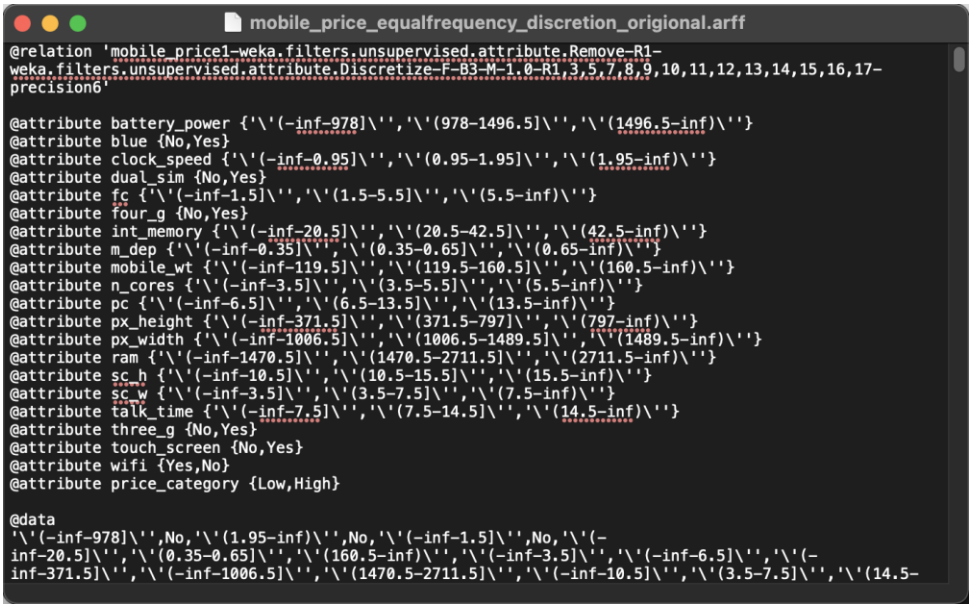


Figure 4 discrete result before rename

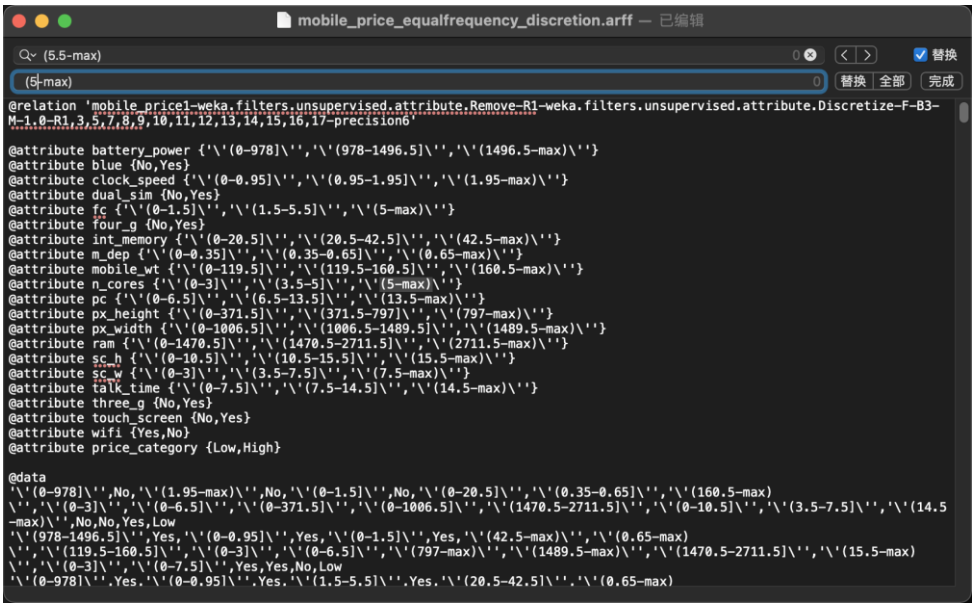


Figure 5 discrete result after renaming

In Task2, we select all the attributes for Association rule mining because all the attributes may affect whether the phone is high or not. After checking all attributes, there is no missing value attribute.

On the Weka Associate page, try setting the Apriori parameters to True for Car, 0.5 for Lower Bound Min Support, 0.5 for MinMetric, and 1000 for NumRules and observing the output.

After clicking Start button, the output shows the error message "No large itemsets and rules found!" as figure shows below, this may cause by higher LowerBoundMinSupport and MinMetric value, so we gradually reduce the parameters and compare the results.

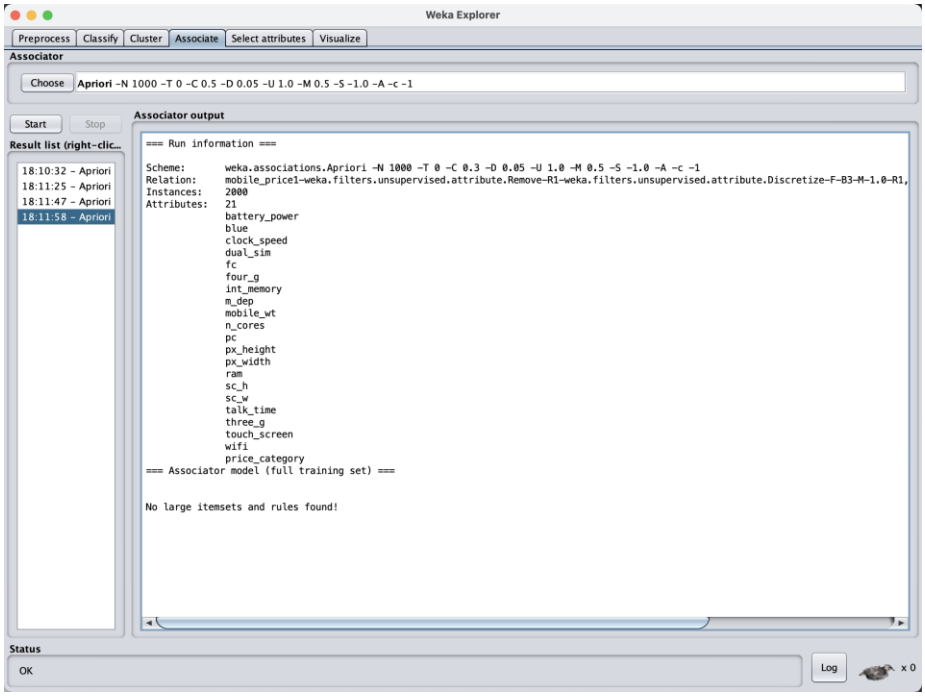


Figure 6 No rules found

Only 14 best rules were found when Lowerboundminsupport = 0.4 Minmetric = 0.5 and all related to low-price mobiles.

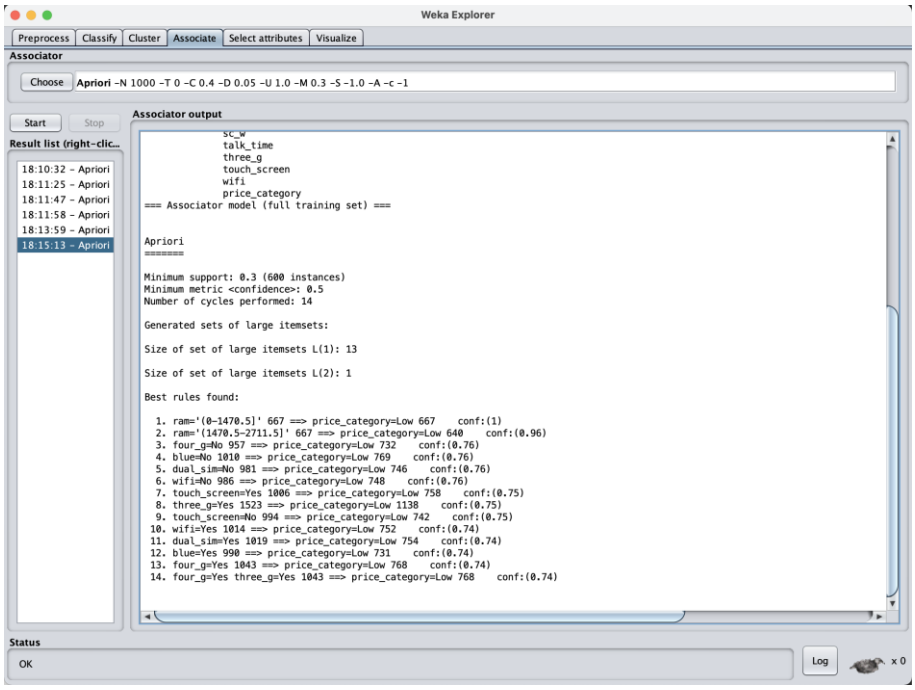


Figure 7 Loweboundminsup = 0.4 Minmeric = 0.5

We keep tweaking the value of Lowerboudsupport and Minmatrix.

When Loweboundsupport is lowered to 0.3, best rules increase significantly and start to get rules associated with higher priced phones. To get more rules, continue to reduce the value of Lowerboundsupport.

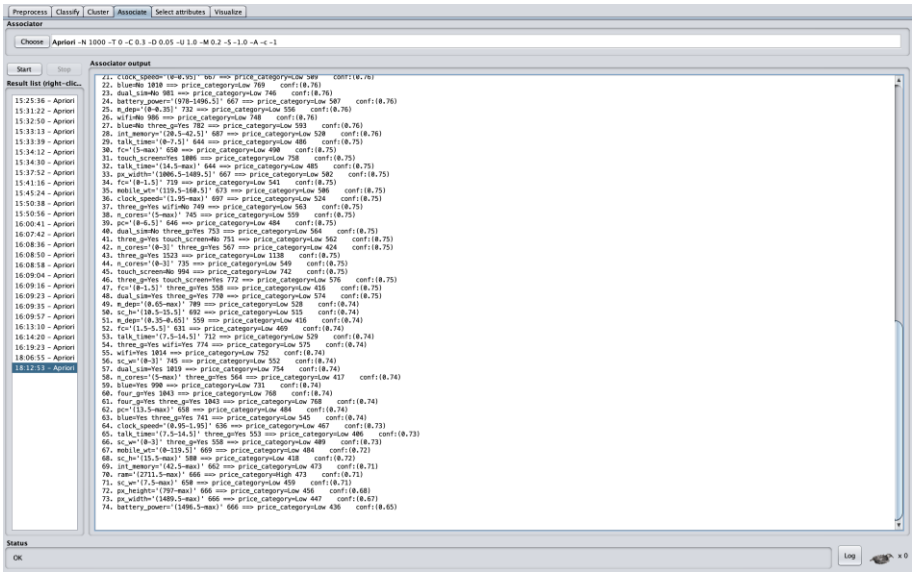


Figure 8 Loweboundminsup = 0.3 Minmeric = 0.5

After multiple parameter adjustments, when Lowerboundminsupport equals 0.1,0.2,0.3,0.4 and Minmetric = 0.2, 29,17 and 14 rules are associated with high-priced mobile phones.

Because Lowerboundminsupport = 0.1 gets more associator outputs, 896 in total and 29 are associated with the high-priced mobile. So, we choose the output results which following this parameter for analysis.

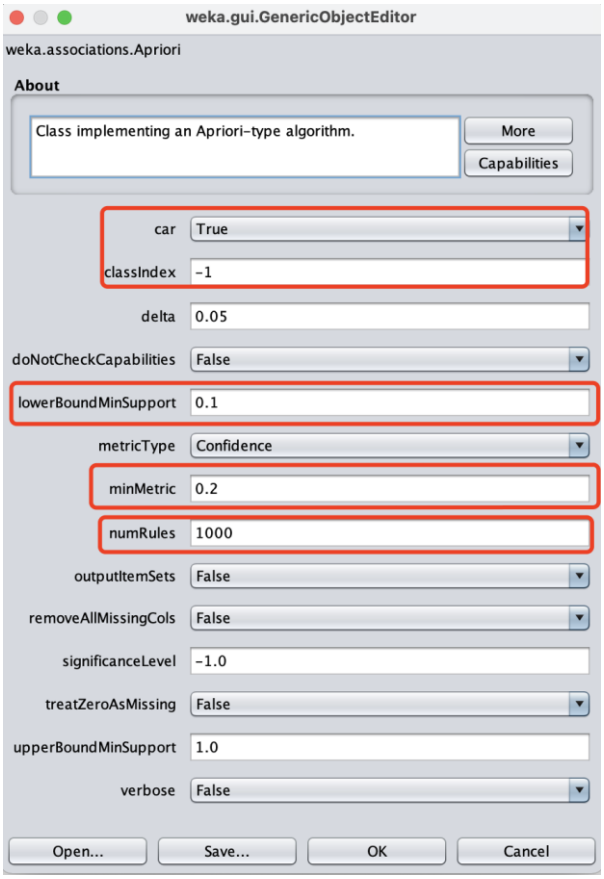


Figure 9 Association rule parameter setting

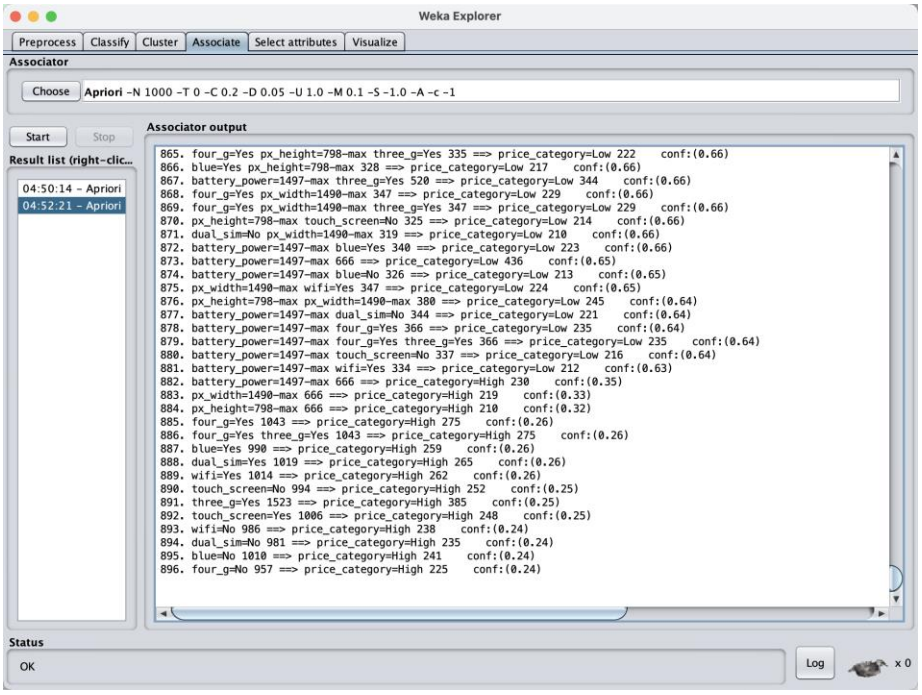


Figure 10 Association rule mining results

3.3 Analysis

According to project requirements for companies that are willing to design high priced phones, we will explain 10 rules according to the rule conference, eight for high priced phones and two for low priced phones. The reason for explaining the low price label rule is to look for features in low price mobiles that provide features that need to be avoided in designing high price mobile.

Confidence(conf) refers to the probability that item B will also appear in the rule of item A. Lift reflects how "the occurrence of item set A" changes the occurrence probability of item set B.

Therefore, Calculating the Lift for each rule is more helpful for us to give recommendations.

Rule 1

111. battery_power=1497-max ram=2712-max 227 ==> price_category=High 207 conf:(0.91)

There are 227 records that satisfy battery_power = 1497-max and ram = 2712-max, of which 207 records price_category labels are “High”, and the confidence of this rule is 0.91 (91%).

In other words, 91% of all phones with a battery capacity greater than 1,496 mah (high capacity range) and a ram size greater than 2,712 MB (high capacity range) are high price category label mobile, because the confidence of this rule is 0.91.

According to formula of Lift,

$$lift(A,B) = \frac{0.91}{0.25} = 3.64$$

Because lift > 1 , then occurrence of battery_power=1497-max ram=2712-max are positively occurrence with price_category=High.

Figure 11 Rule No. 111

Rule 2

531. four_g=Yes ram=2712-max 351 ==> price_category=High 263 conf:(0.75)

There are 351 records that satisfy four_g =Yes and ram = 2712 - max, of which 263 records price_category labels are “High”, and the confidence of this rule is 0.75 (75%).

In other words, 75% of all phones with 4G function and a ram size greater than 2,712 MB (high capacity range) are high price category label mobile, because the confidence of this rule is 0.75.

According to formula of Lift,

$$lift(A,B) = \frac{0.75}{0.25} = 3$$

Because lift > 1, then occurrence of four_g =Yes ram=2712-max are positively occurrence with price_category=High.

Figure 12 Rule No. 531

Rule 3

532. four_g=Yes ram=2712-max three_g=Yes 351 ==> price_category=High 263 conf:(0.75)

There are 351 records that satisfy four_g=Yes ram=2712-max and three_g=Yes, of which 263 records price_category labels are “High”, and the confidence of this rule is 0.75 (75%).

In other words, 75% of all phones with both 3G and 4G functions and the ram size greater than 2,712 MB (high capacity range) are high price category label mobile, because the confidence of this rule is 0.75.

According to formula of Lift,

$$lift(A,B) = \frac{0.75}{0.25} = 3$$

Because lift $>$ 1, then occurrence of four_g =Yes ram=2712-max three_g = yes are positively occurrence with price_category=High.

Figure 13 Rule No. 532

Rule 4

719. ram=2712-max touch_screen=Yes 326 ==> price_category=High 238 conf:(0.73)

There are 326 records that satisfy ram=2712-max and touch_screen=Yes, of which 238 records price_category labels are "High", and the confidence of this rule is 0.73 (73%).

In other words, 73% of all phones with touch screen and the ram size greater than 2,712 MB (high capacity range) are high price category label mobile, because the confidence of this rule is 0.73.

According to formula of Lift,

$$lift(A,B) = \frac{0.73}{0.25} = 2.92$$

Because lift $>$ 1, then occurrence of ram=2712-max touch_screen=Yes are positively occurrence with price_category=High.

Figure 14 Rule No. 719

Rule 5

755. blue=Yes ram=2712-max 338 ==> price_category=High 245 conf:(0.72)

There are 338 records that satisfy ram=2712-max and blue =Yes, of which 245 records price_category labels are "High", and the confidence of this rule is 0.72 (72%).

In other words, 72% of all phones with bluetooth and the ram size greater than 2,712 MB (high capacity range) are high price category label mobile, because the confidence of this rule is 0.72.

According to formula of Lift,

$$lift(A,B) = \frac{0.72}{0.25} = 2.88$$

Because lift > 1, then occurrence of blue=Yes ram=2712-max are positively occurrence with price_category=High.

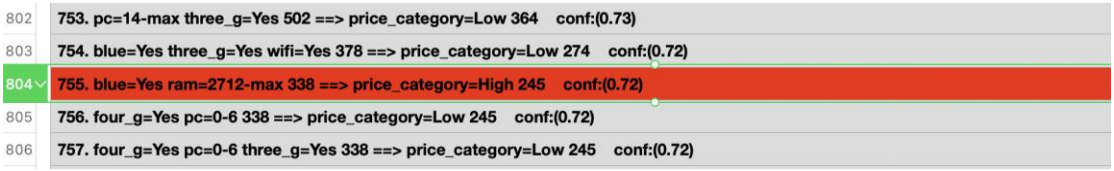


Figure 15 Rule No. 755

Rule 6

767. ram=2712-max three_g=Yes 508 ==> price_category=High 367 conf:(0.72)

There are 508 records that satisfy ram=2712-max and three_g =Yes, of which 367 records price_category labels are “High”, and the confidence of this rule is 0.72 (72%).

In other words, 72% of all phones with 3G function and the ram size greater than 2,712 MB (high capacity range) are high price category label mobile, because the confidence of this rule is 0.72.

According to formula of Lift,

$$lift(A,B) = \frac{0.72}{0.25} = 2.88$$

Because lift > 1, then occurrence of ram=2712-max three_g=Yes are positively occurrence with price_category = High.

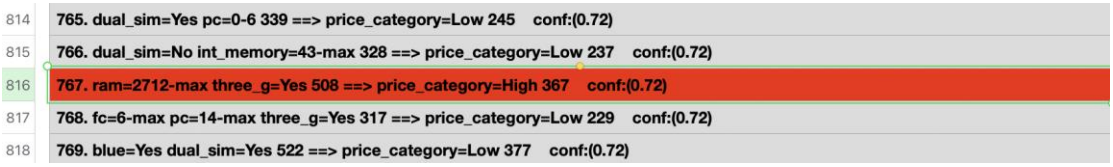


Figure 16 Rule No.767

Rule 7

784. ram=2712-max wifi=No 313 ==> price_category=High 225 conf:(0.72)

There are 313 records that satisfy ram=2712-max and wifi =No, of which 225 records price_category labels are “High”, and the confidence of this rule is 0.72 (72%).

In other words, 72% of all phones without WiFi function and the ram size greater than 2,712 MB (high capacity range) are high price category label mobile, because the confidence of this rule is 0.72.

According to formula of Lift,

$$lift(A,B) = \frac{0.72}{0.25} = 2.88$$

Because lift > 1, then occurrence of ram=2712-max wifi= No are positively occurrence with price_category = High.

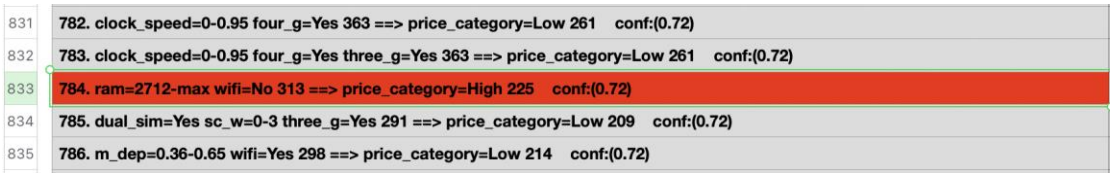


Figure 17 Rule No. 784

Rule 8

810. ram='(2711.5-max)' 666 ==> price_category=High 473 conf:(0.71)

There are 666 records that satisfy ram=2712-max, of which 475 records price_category labels are "High", and the confidence of this rule is 0.72 (71%).

In other words, 71% of all phones with the ram size greater than 2,712 MB (high capacity range) are high price category label mobile, because the confidence of this rule is 0.71.

According to formula of Lift,

$$lift(A,B) = \frac{0.71}{0.25} = 2.84$$

Because lift > 1, then occurrence of ram=2712-max is positively occurrence with price_category = High.

Figure 18 Rule No.810

Rule 9 (low)

1. ram=0-1470 667 ==> price_category=Low 667 conf:(1)

There are 667 records that satisfy ram=0-1470, of which 667 records price_category labels are "Low", and the confidence of this rule is 1 (100%).

In other words, all phones with the ram size less than or equal to 1,470 MB (low capacity range) are low price category label mobile, because the confidence of this rule is 1.

According to formula of Lift,

$$lift(A,B) = \frac{1}{0.75} = 1.33$$

Because lift $>$ 1, then occurrence of ram=0-1470 is positively occurrence with price_category = Low.

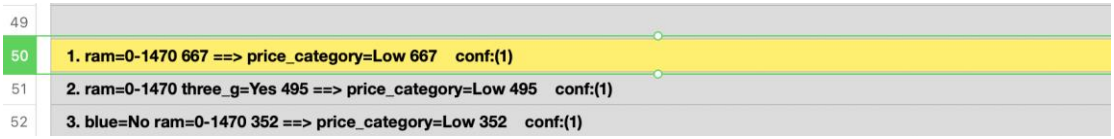


Figure 19 Rule No. 1

Rule 10

8. ram=0-1470 wifi=No 336 ==> price_category=Low 336 conf:(1)

There are 336 records that satisfy ram=0-1470 and wifi=No, of which 667 records price_category labels are “Low”, and the confidence of this rule is 1 (100%).

In other words, all phones without WiFi function and the ram size less than or equal to 1,470 MB (low capacity range) are low price category label mobile, because the confidence of this rule is 1.

According to formula of Lift,

$$lift(A,B) = \frac{1}{0.75} = 1.33$$

Because lift $>$ 1, then occurrence of ram=0-1470 wifi=No are positively occurrence with price_category = Low.

In conclusion, according to Associate Rule 8, our advice to companies designing high-priced phones is that the RAM capacity of the phone should be in the 2712-max range. Company also should avoid to design mobile with a low capacity RAM(0-1,470 MB).

The reason is that in Rule 8, the item set A contains only one attribute RAM =2712-max, and the rule's lift is 2.84. It can fully explain that this attribute has a significant impact on high-priced mobile phones.

In addition, according to Associate Rule 1, if the phone has 0-1,470 MB of RAM, it is possible to design a low price lable mobile. Because in Rule 8, the item set A contains only one attribute RAM = 0-1,470, the item set B only contains price_category = Low, and the rule's lift is 1.33. It can

significantly shows that this attribute has a positive impact on low-cost mobile phones.

4 Task 3 - Classification

Classification refers to a process of categorizing a data set into classes. A model based on training data set will be constructed in classification, which is used to predict class labels or classifying new data.

Classification is one of the supervised learning methods. In this section, we will introduce 3 classification methods and implement them on the mobile phone data set.

4.1 Methodology

In this project, we use Weka to train three different types of classifiers. The following is a brief introduction of these three classification types.

Decision Tree

A decision tree is a structure like a flow chart. Each inner node represents an attribute check, each branch represents the result of the check, and each leaf node represents a class label (decision counts all attributes). The path from root to leaf represents the classification rule[5]. The decision tree consists of the following elements:

- Root node: The complete set containing the sample
- Internal node: corresponding feature attribute test
- : Represents the result of a decision

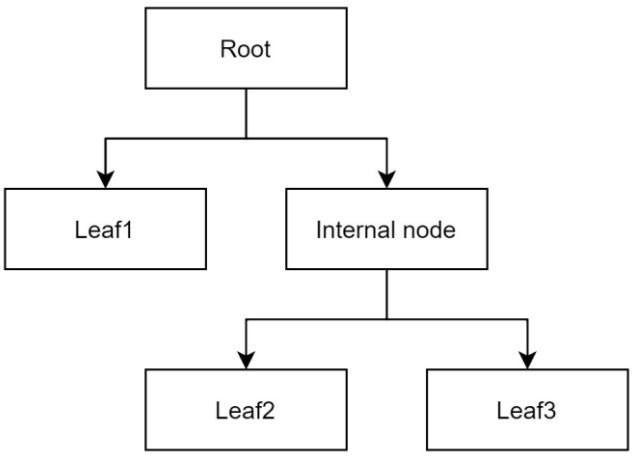


Figure 20 Structure of a decision tree

During the prediction, a certain attribute value is used to make a judgment at the internal node of the tree and decide which branch node to enter according to the result. Once the leaf node is reached, the classification

result is obtained. Decision tree is a supervised learning algorithm based on if-then-else rules [6]. The rules of the decision tree are obtained through training, rather than being made manually. Decision tree is one of the simplest machine learning algorithms [7]. It is easy to implement and highly interpretable.

SVM

SVM is short for Support Vector Machine. It is a supervised learning model and associated learning algorithm for analyzing data in classification and regression analysis[8]. It uses a nonlinear mapping to transform the original training data into a higher dimension if required. In the data space, SVM can find a hyperplane using support vectors and margins [9].

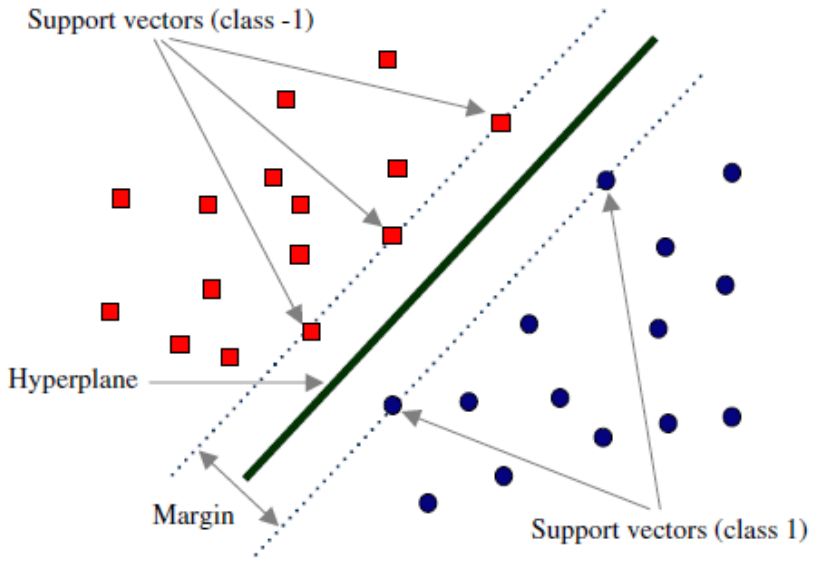


Figure 21 Support Vector Machines [10]

A hyperplane is determined by the normal vector W and the intercept b, and its equation is $X^T W + b = 0$. It can be specified that the side to which the normal vector points is positive class, and the other side is negative class.

In order to find the maximum spaced hyperplane, we can first select two parallel hyperplanes that separate the two types of data, making the distance between them as large as possible. The region within the range of these two hyperplanes is called "margin", and the largest hyperplane is the hyperplane located right in the middle of them. This process is shown in the figure above.

In the case of linearly separability, the data point in the sample points of the training data set that is closest to the separation hyperplane is called the

support vector. Only the support vector plays a role in determining the optimal hyperplane, and no other data points play a role.

Through consulting materials, we found that SVM works very well with higher-dimensional datasets. The SVM algorithm is very stable. Minor changes in the data do not reflect great changes in the results. [10]. In the mobile date set, there is 20 attributes, which is a 20-dimension data set. So we do not need to do dimension reduction. The SVM algorithm is very stable. Minor changes in the data do not reflect great changes in the results.

Naive Bayes classifier

Naive Bayes classifier is one of the Bayes classifiers [11]. Bayes classification algorithm is a classification method of statistics, it is a kind of classification algorithm using probability and statistics knowledge.

$P(A|B)$ indicates the probability that event A will happen, given that event B has already happened. To get $P(B|A)$, we can use the Bayes' theorem:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

The basic idea of Naïve Bayes is, for the given items, calculate the occurrence probability of each category. Then the highest probability determines which category the item belongs to.

The formal definition of Naive Bayes classification is as follows:

1. For given $x = \{a_1, a_2, \dots, a_m\}$, x is an item to be classified, a_m is an attribute of x.
2. $C = \{y_1, y_2, \dots, y_n\}$, which is a set of classes.
3. Calculate $P(y_1|x), P(y_2|x), \dots, P(y_n|x)$
4. If $P(y_k|x) = \max\{P(y_1|x), P(y_2|x), \dots, P(y_n|x)\}$, then $x \in y_k$.

The whole Naive Bayesian classification is divided into three stages:

The first stage: preparation stage, the task of this stage is to determine the feature attributes according to the specific situation and classify each feature attribute appropriately. Then manually classify a part of the items to be classified to form a training sample set. The quality of classifier is largely determined by feature attributes, feature attribute division and training sample quality.

The second stage: classifier training stage. The task of this stage is to generate a classifier. The main work is to calculate the occurrence frequency of each category in the training sample, and also calculate the conditional probability estimate of each category for each feature attribute division.

The third stage: the application stage. The task of this stage is to use the classifier to classify the classified items. The input is the classifier and the items to be classified, and the output is the mapping relationship between items and category.

4.2 Implementation in Weka Analysis

In this project, precision and recall of the predicted results will be mainly considered to train different classifiers based on different machine learning algorithms and make evaluations. This choice is only for our project. In real life, the evaluation indicators need to be determined according to the project background and specific requirements.

Cross-validation is used to evaluate the prediction performance of the model, especially the performance of the trained model on the new data, which can reduce overfitting to a certain extent. In the project, we used 10-fold cross validation.

4.2.1.1 Decision Tree

The data used to model the J48 decision tree does not need to be discrete[12], so we imported the data set, which was cleaned in task 1 into Weka. We are using all data and all attributes to construct a J48 decision tree to observe the result.

According to the prediction results of the model and the visualized decision tree, we find that the maximum depth is 7 and the width of the decision tree under this condition is 77. If the training data set is extensive, the maximum width of the decision tree could be 2^7 ,which is 512. Moreover, the prediction recall of the model using all attributes for high-priced mobile can reach 98.2%.

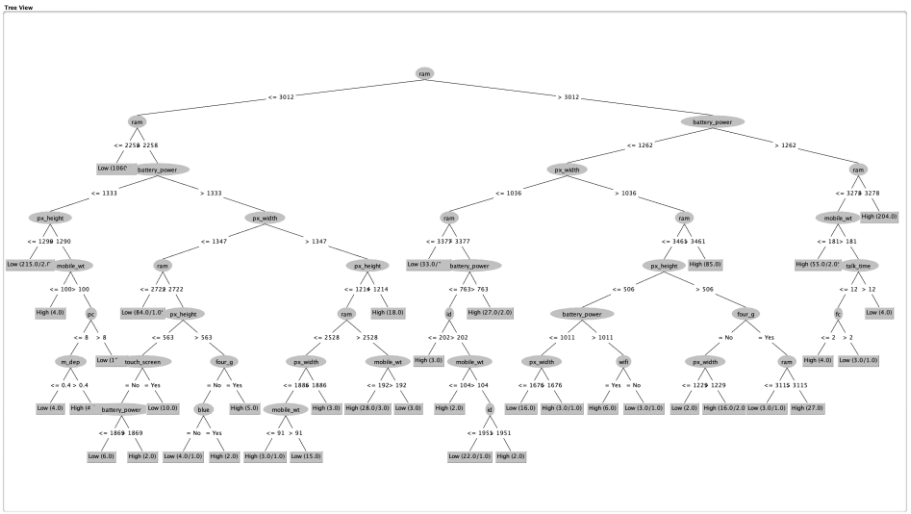


Figure 22 Visualized Decision tree with original data

```
Number of Leaves  :     39

Size of the tree :      77


Time taken to build model: 0.02 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances        1979               98.95   %
Incorrectly Classified Instances        21                1.05   %
Kappa statistic                          0.9721
Mean absolute error                      0.0179
Root mean squared error                  0.0947
Relative absolute error                  4.781  %
Root relative squared error             21.8692 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.992     0.018     0.994      0.992     0.993      0.972    0.998     0.999     Low
               0.982     0.008     0.976      0.982     0.979      0.972    0.998     0.995     High
Weighted Avg.  0.990     0.015     0.990      0.990     0.990      0.972    0.998     0.998

=== Confusion Matrix ===

    a    b   <-- classified as
 1488   12 |    a = Low
    9  491 |    b = High
```

Figure 23 Results of all data as training sets

In order to evaluate the model, we are using the 10 cross-validation method of Weka for evaluation.

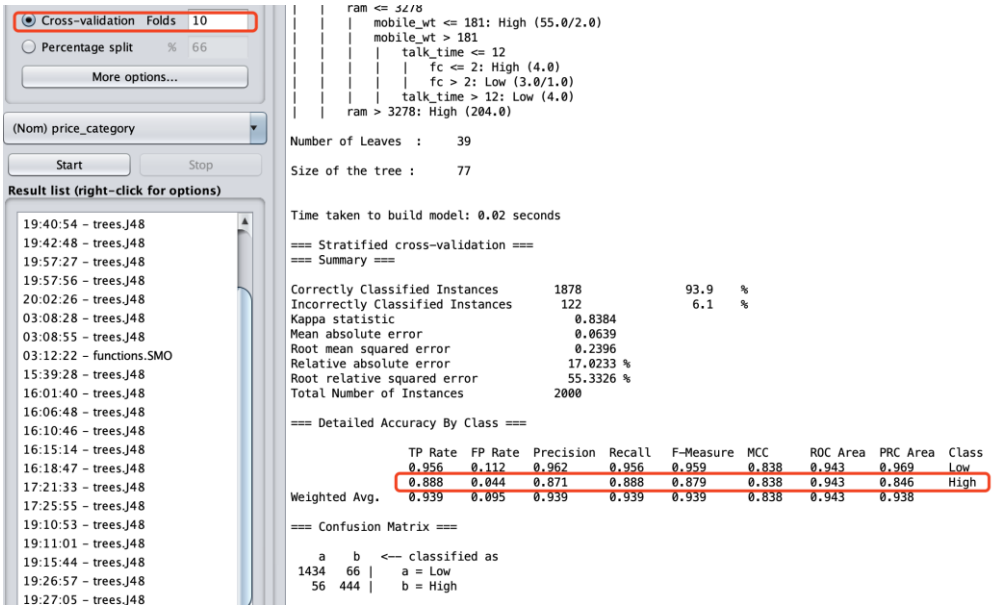


Figure 24 10-fold cross-validation results

The verification results show a significant decrease in the prediction recall of high-priced label mobile, with only 88.8% of high-priced mobile being correctly predicted, while the prediction recall of high-priced mobiles is not high. After analyzing the data set, to simplify the readability of the decision tree results, three new attributes are added, which are screen size [13], pixel size and PPI [14]. We try to reduce some attributes to improve the recall of model prediction.

Because the J48 decision tree is based on the C4.5 algorithm [15] to calculate the information gain ratio of each attribute to determine the root

node, we are using GainRatioAttributeEval algorithm to evaluates the gain ratio of the attribute with respect to the class.

According to select attribute result, attribute ranked result is 13,1,12,15,6,2,4,19,17,18,16,11,5,3,14,7,8,9,10 : 19

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

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 20 price_category):
        Gain Ratio feature evaluator

Ranked attributes:
 0.2804926  13 ram
 0.0261579   1 battery_power
 0.0195144  12 px_size
 0.0087653  15 ppi
 0.0007849   6 four_g
 0.0005089   2 blue
 0.0004047   4 dual_sim
 0.0002782  19 wifi
 0.0001214  17 three_g
 0.0000471  18 touch_screen
 0          16 talk_time
 0          11 pc
 0           5 fc
 0           3 clock_speed
 0          14 sc_size
 0           7 int_memory
 0           8 m_dep
 0           9 mobile_wt
 0          10 n_cores

Selected attributes: 13,1,12,15,6,2,4,19,17,18,16,11,5,3,14,7,8,9,10 : 19
```

Figure 25 Ranked attributes result

We first removed the last nine attributes that no value displayed to the gain ratio of the class：

- 16 talk_time
- 11 pc
- 5 fc
- 3 clock_speed
- 14 sc_size
- 7 int_memory
- 8 m_dep
- 9 mobile_wt
- 10 n_cores

The J48 decision tree prediction model was established again with the data after removing some attributes. When all the data were used as the training set, the width of the new decision tree generated is still 53, and the maximum depth is 8 layers. The recall of the prediction result of high-price mobiles decreased slightly (to 97.2%), but the recall of the 10-fold cross-

validation results significantly increased to 90.6% compared with the previous one.

```
Number of Leaves  :     27

Size of the tree :      53


Time taken to build model: 0.01 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances        1970               98.5    %
Incorrectly Classified Instances        30                1.5    %
Kappa statistic                          0.9601
Mean absolute error                      0.0263
Root mean squared error                  0.1147
Relative absolute error                  7.0189 %
Root relative squared error             26.4977 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.989    0.028    0.991      0.989    0.990      0.960    0.997     0.999     Low
                 0.972    0.011    0.968      0.972    0.970      0.960    0.997     0.990     High
Weighted Avg.    0.985    0.024    0.985      0.985    0.985      0.960    0.997     0.996

=== Confusion Matrix ===

    a    b   <-- classified as
 1484   16 |    a = Low
   14  486 |    b = High
```

Figure 26 Results of all data as training sets

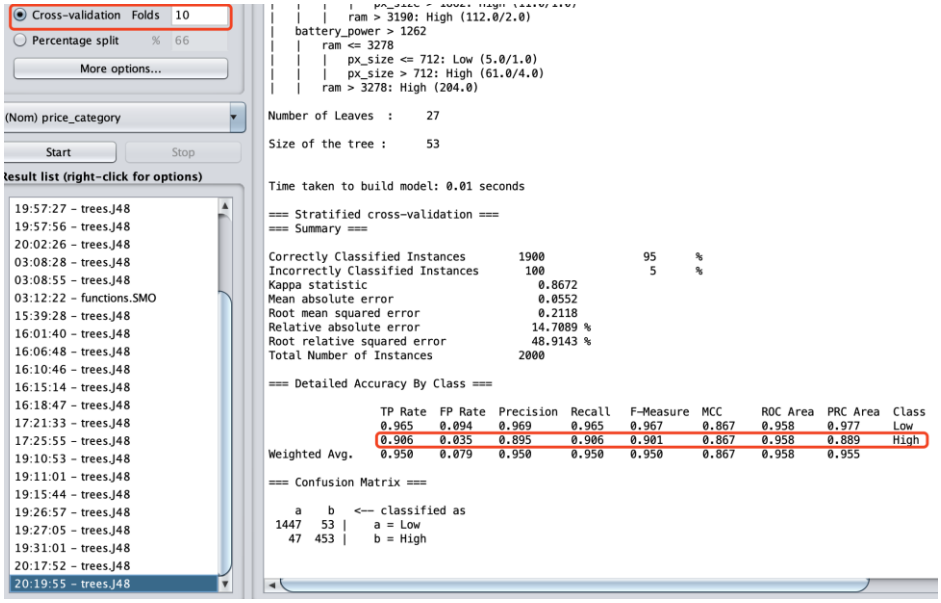


Figure 27 10-fold cross-validation results

Continue to GainRatioAttributeEval for the remaining attributes and continue to reduce attribute observations. Thus, remove touch_screen and sc_size attributes that have a value less than 0.0001 relatives to the gain ratio of the class

We use the new data set to generate the J48 decision tree. By observing the output results, we find that the width of the new decision tree is still 53, the maximum depth is 8 layers, and the visualised decision tree has no change compared with the original one.

When all the data are used as the training set, the recall of high-priced mobile phones is still 97.2%, but the results of ten-fold cross-validation show that the recall of high-priced mobile phones continued to rise slightly to 91% in Figure 29. This indicates that the remotion of touch_screen and sc_size attributes can improve the high price mobile prediction recall and precision of the model for this model.

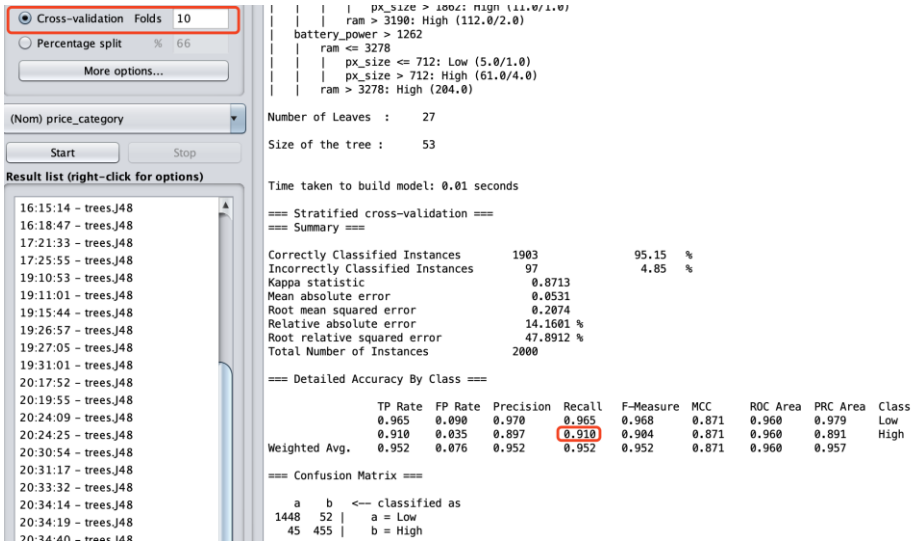


Figure 28 10-fold cross-validation results

If we continue to reduce the attributes, the recall of the prediction result for the high price category mobiles begins to decrease.

We decided to use these nine attributes in the data set to construct the J48 decision tree model according to the above process.

- 0.280493 6 ram
- 0.026158 1 battery_power
- 0.019514 5 px_size
- 0.008765 7 ppi
- 0.000785 4 four_g
- 0.000509 2 blue
- 0.000405 3 dual_sim
- 0.000278 9 wifi
- 0.000121 8 three_g
-

4.2.1.2 Result Analyses for Decision Tree

For the prediction model of our project, it is easy to obtain some attributes that have a significant impact on the results by observing the visual decision tree (the figure below). Take one of the J48 decision tree

prediction rule about high-priced mobile phones as an example to explain. Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction.

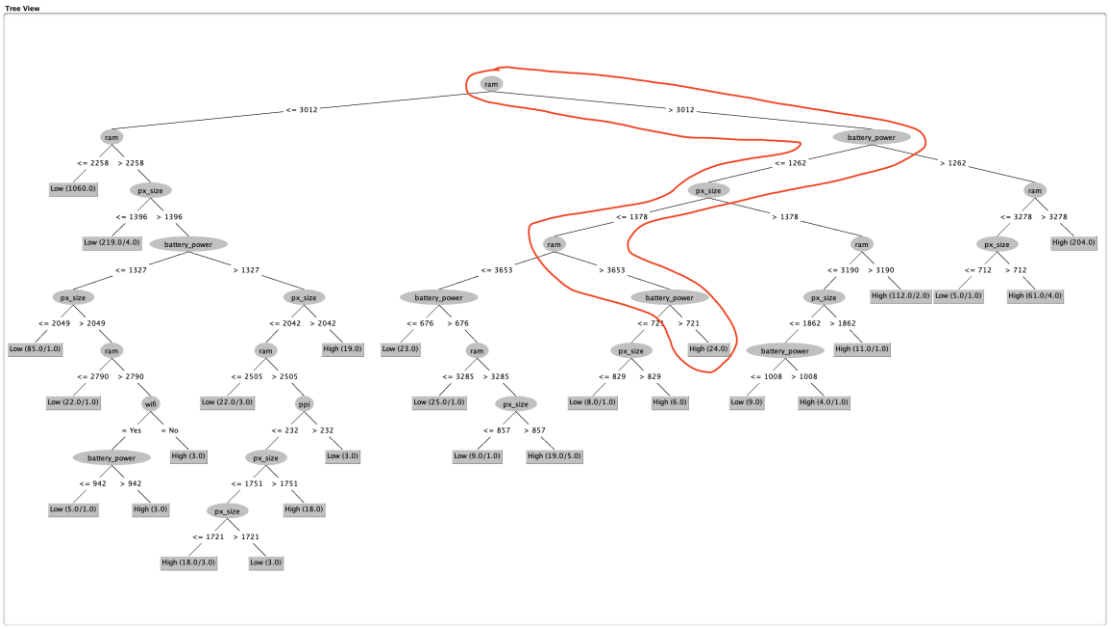


Figure 29 Example path of decision tree

```
ram > 3012
|   battery_power <= 1262
|   |   px_size <= 1378
|   |   |   ram <= 3653
|   |   |   |   battery_power <= 676: Low (23.0)
|   |   |   |   battery_power > 676
|   |   |   |   |   ram <= 3285: Low (25.0/1.0)
|   |   |   |   |   ram > 3285
|   |   |   |   |   |   px_size <= 857: Low (9.0/1.0)
|   |   |   |   |   |   px_size > 857: High (19.0/5.0)
|   |   |   ram > 3653
|   |   |   |   battery_power <= 721
|   |   |   |   |   px_size <= 829: Low (8.0/1.0)
|   |   |   |   |   px_size > 829: High (6.0)
|   |   |   |   battery_power > 721: High (24.0)
```

Figure 30 One decision tree prediction rule

Take one of the J48 decision tree prediction rule about high-priced mobile phones as an example to explain (shown in Figure 31). Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction.

Based on this prediction rule, the model successful predicted 24 cases of high-priced mobile phones without failure。

IF ram > 3012

AND battery_power <=1262

AND px_size <=1372

AND ram >3653

AND battery_power > 721

THEN pricr_category = high

In conclusion, when using the data provided for the project to predict high-priced mobile phones, the J48 decision tree model we constructed predicts that 507 mobile phones have high price labels, and the real high price label is 455, the recall is 91%. The overall predicted recall of the example is 95.2%. Although the overall predicted recall of other models is higher than the model we selected after trying, we chose the model with the highest predicted recall within the high price category. The project aims to design high-priced mobile phones.

4.2.2.1 SVM

SVM is good at processing numerical and high dimension data, so there is no need for reduction or dispersion [16].

Import the data in WEKA that has only been processed by TAKS 1. Click the "Choose" button and select "SMO" under the "function" group.

When using all the data as the training set to build the model, the recall of high price label prediction result of high-priced mobile phones is 99%

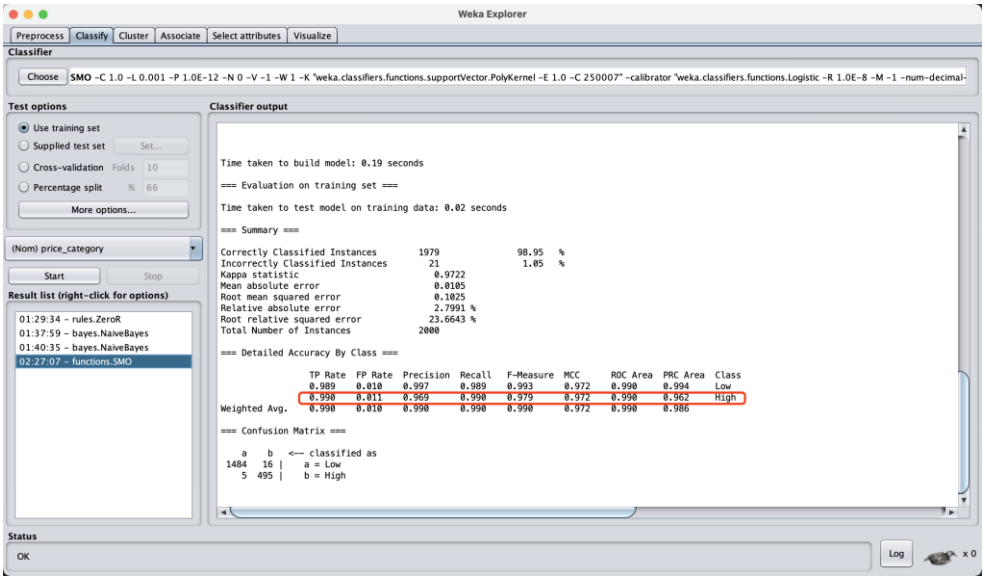


Figure 31 Using all data as the training set to SVM

We evaluate the SVM model by using a 10-fold cross-validation approach. The results show that the overall prediction precision of the SVM model is 98%, the overall prediction recall is 98%, and the precision for high-priced mobile phones is 95.3%. That is to say, 95.3% of all the instances classified as high price mobile phones are real high-priced mobile phones.

The recall rate is 96.8%, which means that the SVM model correctly predicted 96.8% of all high-priced phones in the data set, which is a very good result.

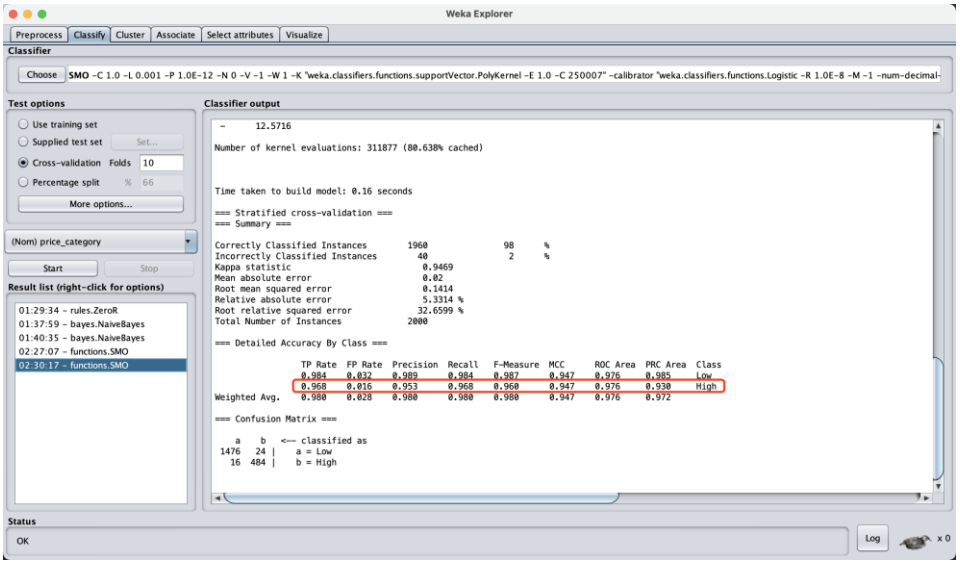


Figure 32 10-fold cross-validation to SVM

4.2.2.2 Result Analyses for SVM

Compared with the prediction results made by the decision tree classifier, the results obtained by using SVM are higher. However, because SVM is based on mathematical, the results are difficult to interpret. The predicted model results can only be obtained without knowing what the prediction criteria are. We can't get the information of what attribute will critically impact high price mobile design from the SVM model

Thus, we tried to use a naive Bayes classifier

4.2.3.1 Naïve Bayes

According to the Naive Bayes prediction principle, conditional class independence is required; otherwise, it will cause a loss of accuracy. Since the data used in our project has numerical variables, the hypothesis of normal distribution needs to be satisfied.

Since the data in the *mobile_price.csv* is processed and not properly normal distributed, errors may occur in the predicted results. We discretize data to reduce the error caused by the non-normal distribution of data.

There are two kinds of discretization methods: equidistant and Isofrequency [17].

Equidistant discretization divides the value domain of the attribute from the minimum value to the maximum value into n bins with the same distance. It tends to distribute the attribute value to each interval unevenly. Isofrequency discretization is to evenly distribute data in n bins to ensure that the data in each bin is equal.

It should be noted that the two discrete methods also have some disadvantages. The disadvantage of equidistant discretization is that it is too sensitive to noise. Some intervals have very many values, while some intervals are very few, which may damage the data model established after discretization. The Isofrequency discretization does not have very large or tiny intervals. However, according to the principle of Isofrequency dispersion, to ensure data consistency in each interval, it is possible to divide the two values that were initially the same into different intervals, which may cause more damage to the prediction model equidistance.

We deal with the original data by equidistant discretization. Because the project data is processed, there are many meaningful extremes value of 0 for px_height and sc_w. Equidistant discretization can effectively reduce the influence of these value and outlier. Although there are still a few errors in the prediction results after the discretization data, the benefits are still worth trying.

To eliminate the relationship between the attributes in the data set and eliminate the relationship between these attributes, we calculated the Kendall correlation coefficient [18] between every two attributes. We deleted one of the two attributes with a strong correlation coefficient. So that all the attributes in the data set used for training were independent.

According to the Kendall correlation coefficient analysis results for our data set, the coefficient between fc and pc is 0.52. There is a weak relationship as the coefficient is greater than 0.3. As a result of the average coefficient value of pc is greater than fc, pc is deleted between these two attributes.

The correlation coefficient between px_height and he px_width is 0.33, leading to the existence of a weak relationship, and the attribute of px_height (0.076) with a higher average coefficient can be deleted.

Similarly, a weak relationship (0.36) between sc_h and sc_w cannot be ignored, and sc_h with a relatively high average coefficient (0.068) is deleted.

Preserve other properties where the correlation coefficient is less than 0.3 or negative. According to the Kendall correlation coefficient, the coefficient is in the range of 0-0.3 can be understood that there is no correlation so that the error can be ignored.

	battery_pow	clock_speed	fc	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	Average
battery_pow	1	0.00660265	0.02438844	-0.0028654	0.0230984	0.00187788	-0.0209728	0.02074209	0.00653056	-0.0058143	-0.0008904	-0.0202116	-0.0185109	0.03645639	
clock_speed	0.00660265	1	-0.00381	0.00374513	-0.0108475	0.0075008	-0.0061234	-0.004388	-0.0090867	-0.0058572	0.00325894	-0.0214656	-0.0104163	-0.0094584	
fc	0.02438844	-0.00381	1	-0.0191899	0.0095427	0.01882629	-0.01148	0.52382524	-0.014535	-0.0065971	0.01379846	-0.0067861	-0.0008475	-0.0005245	
int_memory	-0.0028654	0.00374513	-0.0191899	1	0.00490938	-0.0234073	-0.0201248	-0.022935	-0.00136	-0.0057373	0.02237507	0.02778499	0.01126811	-0.0017361	
m_dep	0.0230984	-0.0108475	0.0095427	0.00490938	1	0.01592528	-0.0038487	0.02015871	0.01870362	0.0162531	-0.0070408	-0.0178428	-0.0144787	0.01212364	
mobile_wt	0.00187788	0.0075008	0.01882629	-0.0234073	0.01592528	1	-0.0138854	0.01274227	0.00791354	0.00066633	-0.0015881	-0.0238638	-0.0137909	0.00452709	
n_cores	-0.0209728	-0.0061234	-0.01148	-0.0201248	-0.0038487	-0.0138854	1	-0.0015101	-0.0032737	0.0174279	0.00378276	0.0003052	0.02096573	0.00953435	
pc	0.02074209	-0.004388	0.52382524	-0.022935	0.02015871	0.01274227	-0.0015101	1	-0.0101384	0.00216608	0.01983433	0.00372847	-0.0249276	0.01013466	0.11067377
px_height	0.00653056	-0.0090867	-0.014535	-0.00136	0.01870362	0.00791354	-0.0032737	-0.0101384	1	0.3328914	-0.0207133	0.03685083	0.02053415	-0.0069918	
px_width	-0.0058143	-0.0058572	-0.0065971	-0.0057373	0.0162531	0.00066633	0.0174279	0.00216608	0.3328914	1	0.00217209	0.01596597	0.01749054	0.004247	0.074
ram	-0.0008904	0.00325894	0.01379846	0.02237507	-0.0070408	-0.0015881	0.00378276	0.01983433	-0.0207133	0.00217209	1	0.01127577	0.01819359	0.00784102	
sc_h	-0.0202116	-0.0214656	-0.0067861	0.02778499	-0.0178428	-0.0238638	0.0003052	0.00372847	0.03685083	0.01596597	0.01127577	1	0.35813815	-0.0125696	
sc_w	-0.0185109	-0.0104163	-0.0008475	0.01126811	-0.0144787	-0.0137909	0.02096573	-0.0249276	0.02053415	0.01749054	0.01819359	0.35813815	1	-0.0154317	0.065
talk_time	0.03645639	-0.0094584	-0.0005245	-0.0017361	0.01212364	0.00452709	0.00953435	0.01013466	-0.0069918	0.004247	0.00784102	-0.0125696	-0.0154317	1	
Average			0.10904365						0.076			0.068			

Figure 33 Kendall correlation coefficient table

Because Kendall's correlation coefficient can only compute consecutive numbers, those attributes values are *yes* or *no* need to determine whether they are independent or not by our life experience. This may cause minor errors, but because the data used is processed, these errors can be ignored.

By subjective judgment, there will be a connection between 3G and 4G. Based on the comparison of the original data set and life experience, a phone that supports 4G functionality must include 3G functionality. So, keep 4G in these two properties and delete 3G.

Remove the properties mentioned above in the WEKA Process. Click the Choose button in Classify and select the Naive Bayes classifier under Bayes.

First, we use all the data as the training set to build the model。

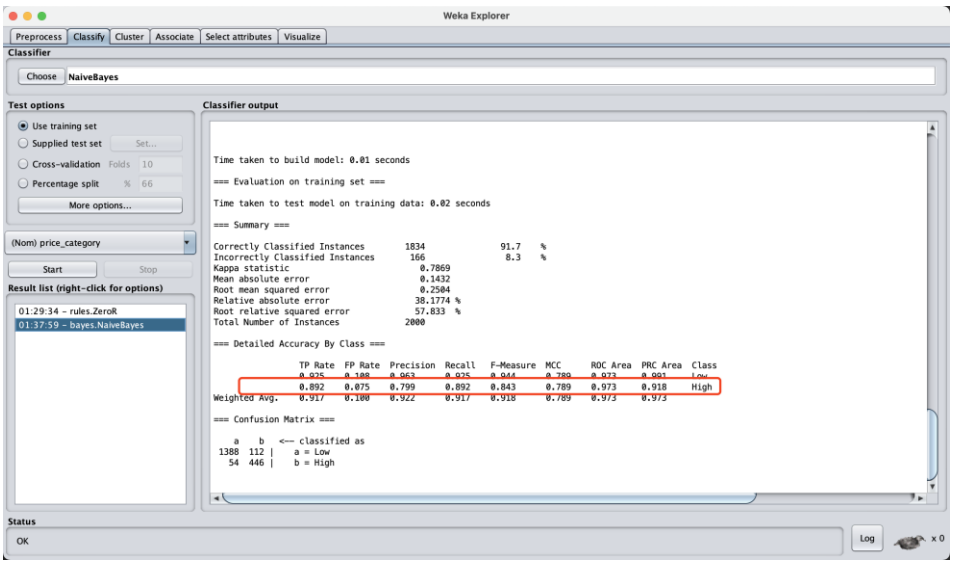


Figure 34 Use all data as training set to Naive Bayes

According to the output, when all the data are used as the training set, the recall of the Naive Bayesian model for the prediction of high-priced mobile phones is 89.2%.

We use 10-fold cross-validation to evaluate the model, and the output was as follows.

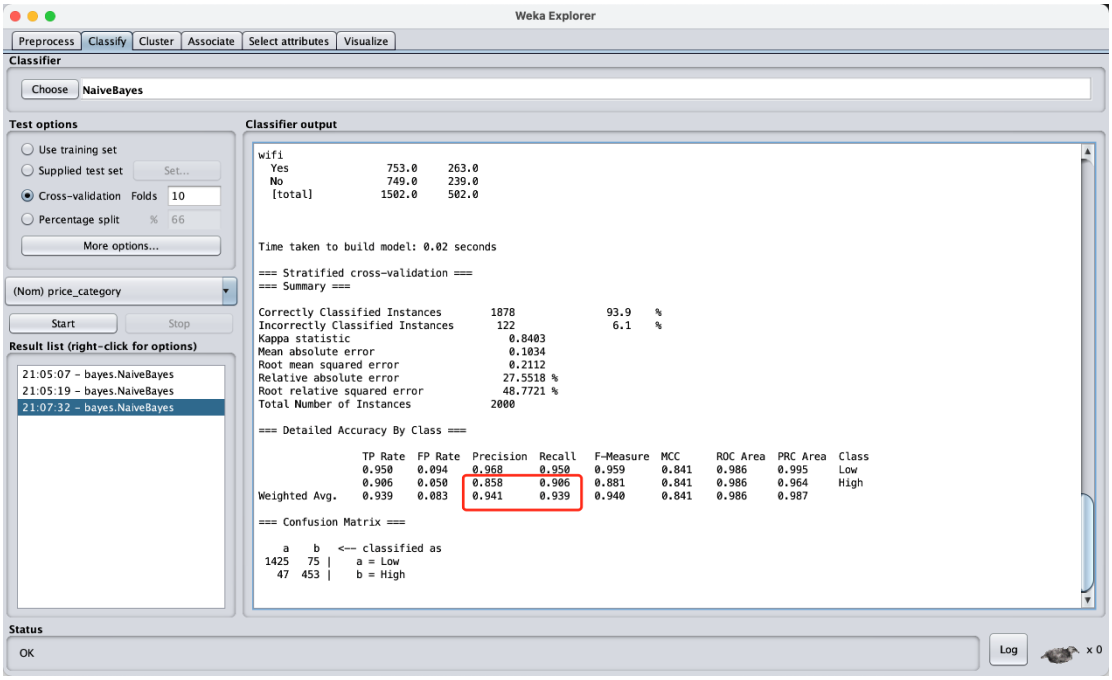


Figure 35 10-fold cross-validation to Naive Bayes

Through ten-fold cross-validation, the overall prediction precision of the Naive Bayes model is 93.9%, and the recall is 91.1%. The prediction precision for high-priced phones is 85.8%, which means that only 85.8% of all the instances that are classified as advanced phones in the results returned after the Naive Bayes prediction are truly advanced phones. The recall for high-price phones was 90.6%, and the Naive Bayes model correctly predicted 90.6% of high-price phones in all instances.

4.2.3.2 Result Analyses for Naïve Bayes

By selecting part of the Naive Bayes classification process, this section analyzes the attributes that influence the design of high-priced mobile phones. Taking the RAM attribute as an example, the ram after equidistant dispersion is divided into three intervals (0-1503.3),(1503.3-2750.7) and (2750.7-max), which can be understood as low, medium and high levels.

$$P(low) = \frac{1}{684} = 0.0015$$

$$P(medium) = \frac{33}{402} = 0.0821$$

$$P(high) = \frac{474}{668} = 0.7216$$

The difference between different levels is evident, indicating that the RAM specification has a noticeable influence on whether the high-priced phone is a high-priced phone. Moreover, since $P(high) = 0.7216$, it means that

the phone with high-capacity RAM is more likely to become a high-priced phone.

```
ram
  '(-inf-1503.333333]'            683.0    1.0
  '(1503.333333-2750.666667]'     639.0   33.0
  '(2750.666667-inf)'             181.0  469.0
  [total]                        1503.0  503.0
```

Figure 36 Ram attribute in Naive Bayes

4.3 Analysis

The precision of the three classifiers for a high-priced label is all good (greater than 50% of random prediction). The highest prediction recall for high-priced mobile phones is the SVM classifier (recall=96.8%), followed by the decision tree model (recall=91%). The worst model prediction result was the Naive Bayes classifier (recall =88%).

Although the SVM model gives the best-predicted result, this project requires advice on designing a high-priced phone, and an SVM model is not recommended. This is because the prediction process of SVM relies on mathematical. Compared with other methods, it isn't easy to interpret the influence of different attributes on the results through the prediction process of SVM. And when the data type is more complex, such as categorical and numerical exist at the same time does not apply to SVM.

Of course, SVM also has some advantages. For example, SVM is good at dealing with small data sets of high dimensions. SVM maps non-linear features to low-dimensional features to high-dimensional features, and uses kernel methods to directly compute the inner product between high-dimensional features. This avoids explicitly computed non-linear feature mappings. You can also do the linear classification of higher-dimensional feature Spaces [19].

Compared with SVM classification, J48 decision tree classification relies on the C4.5 algorithm to calculate the gain ratio of each attribute, and the one with the largest information gain ratio result is regarded as the root node. J48 decision tree has the advantages of simple calculation and strong interpretation, which is more suitable for dealing with samples with missing attribute values and dealing with unrelated features. However, when the data set contains a large number of attributes, if the attributes are not screened, the size of the decision tree will be too wide, and it is difficult to interpret the prediction rules. And every time the item data set adds a

new attribute, it needs to modify the tree structure, and it is easy to over-fitting.

Combined with the data set we used, there are 20 attributes in it. Suppose all attributes are used to construct a decision tree. In that case, the decision tree level will be relatively deep, and the tree structure is not good enough to be readable, so it is difficult to conveniently find out the attributes that impact the design of high-priced mobile phones. Although the decision tree does not need to discretize the data, it needs to rank the attributes according to the attribute information to reduce the attributes to make the data readable. Reducing the number of attributes used to build the model may result in errors in the prediction results.

Although the result of the Naive Bayes prediction model is the lowest for the prediction result of this project, in general, the Naive Bayes classification has many advantages. Compared with other classification methods such as Logistic regression, the Naive Bayes classifier has better performance and requires less training data when the hypothesis independence of variables is valid. Naive Bayes classifiers perform better in multiple classification variables, but for numerical variables, the data needs to satisfy the assumption of normal distribution. Again, Naive Bayes has some limitations. For example, Naive Bayes is the assumption of independent forecasting. The predicted attributes in the model need to be independent and not affect each other. In real life, this is almost impossible, the variables constantly interact. Another disadvantage is that if the category of the classification variable (in the test dataset) is not always observed in the training dataset, then the model will assign a 0 probability to it and will not be able to predict. This is often referred to as the "zero frequency".

In general, for this project, we recommend using the J48 decision tree model for prediction. Compared with the Naive Bayes model, the decision tree model can return a more satisfactory result, and at the same time, it can avoid the problem that the SVM model is difficult to interpret. The decision tree model is a model that can balance out most of the requirements for out project. It is important to note that this conclusion is specific to this project and is not generally representative.

5 Task 4 - Clustering

Clustering is one of the unsupervised learning methods[20]. This part introduces two different clustering methods. One is K-means clustering, and another one is Density Based clustering.

5.1 Methodology

K-Means

K-means is to divide N points (which can be an observation or an instance of the sample) into K clusters so that each point belongs to the cluster corresponding to the nearest mean value (this is the cluster center), and take it as the clustering standard.

The most common way is using iteration[21]. The algorithm is performed alternately in the steps:

1. First, initialize k samples as the initial clustering centres.
   $a = a_1, a_2, \ldots, a_k$
2. For each sample in the data set x_i, calculate its distance to k clustering centres and divide it into the class corresponding to the clustering centre with the smallest distance.
   $$d(i,j) = \sqrt[q]{|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \cdots + |x_{ip} - x_{jp}|^q}$$
   When q=1, d is Manhattan Distance, which means the sum of the distances in each dimension.
   When q=2, d is Euclidean Distance, which means the length of a line segment between the two points.
3. For each category a_j, recalculate its clustering centre
   $$a_j = \frac{1}{|c_i|} \sum_{x \in c_i} x$$
4. Repeat steps 2 and 3 above until a specific stop condition (number of iterations, minimum error change, etc.) is reached.

Density Based cluster

Density Based clustering is an unsupervised learning method. It can identify the different groups/clusters in the data, and cluster the contiguous regions with high point density in the data space, and separate other such clusters by the contiguous regions with low point density[22].

There are two parameters in Density Based clustering, Eps and MinPts. Eps is the maximum radius of the neighborhood, MinPts is the minimum number of points in an Eps-neighborhood of that point.

To do Density Based cluster, first is to select a point p arbitrarily. Then retrieve all density-reachable points from p, with respect to Eps and MinPts. If p is a border point, the algorithm will visit the next point of the database. If p is a core point, then a cluster is formed. Repeat the above step until all the data points have been processed.

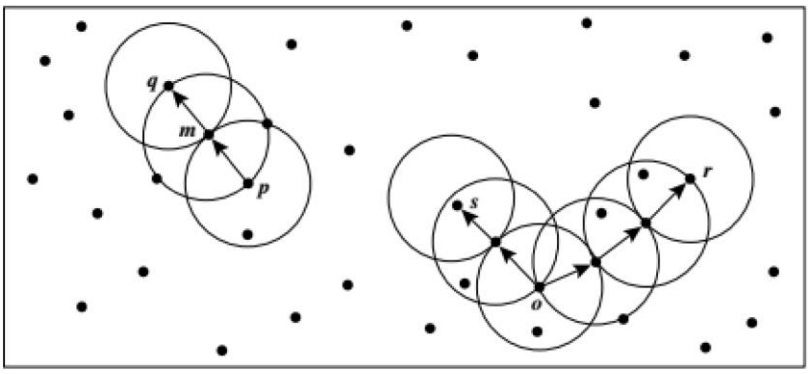


Figure 37 Density-Reachable and Density-Connected

A cluster satisfies two attributes: 1). All the points in the cluster are density-connected to each other. 2). If a point is identifiable from a point in the cluster, it is also part of the cluster [23].

5.2 Clustering in Weka

The data sheet has 22 columns. K-means in Weka works best for numerical values. The id column makes no sense for clustering so that it would be removed. For clustering work, there are 14 numeric columns and 6 categorical or binary columns. First, use the original data to do K-means clustering.

Before doing clustering, the target class need to convert to nominal. In Weka, use NumericToNominal to convert the price_category into nominal form.

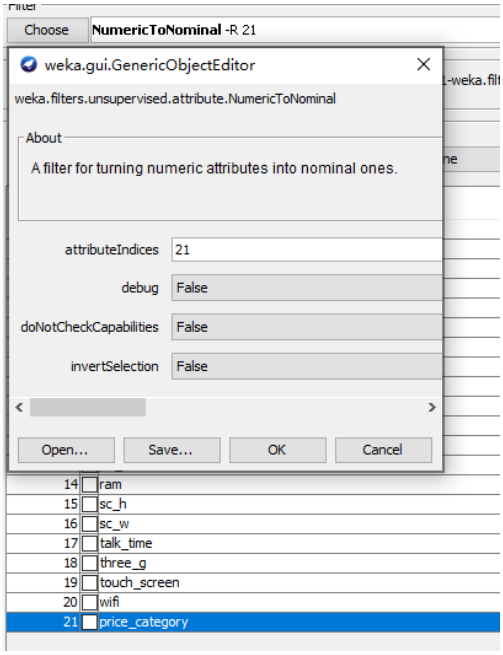


Figure 38 Convert price_category to nominal

Then do the clustering, the result is shown as following. The final incorrectly clustered instances are 994, take 49.7% in total. It is a terrible result, which seems to be clustering randomly, nearly 50%.

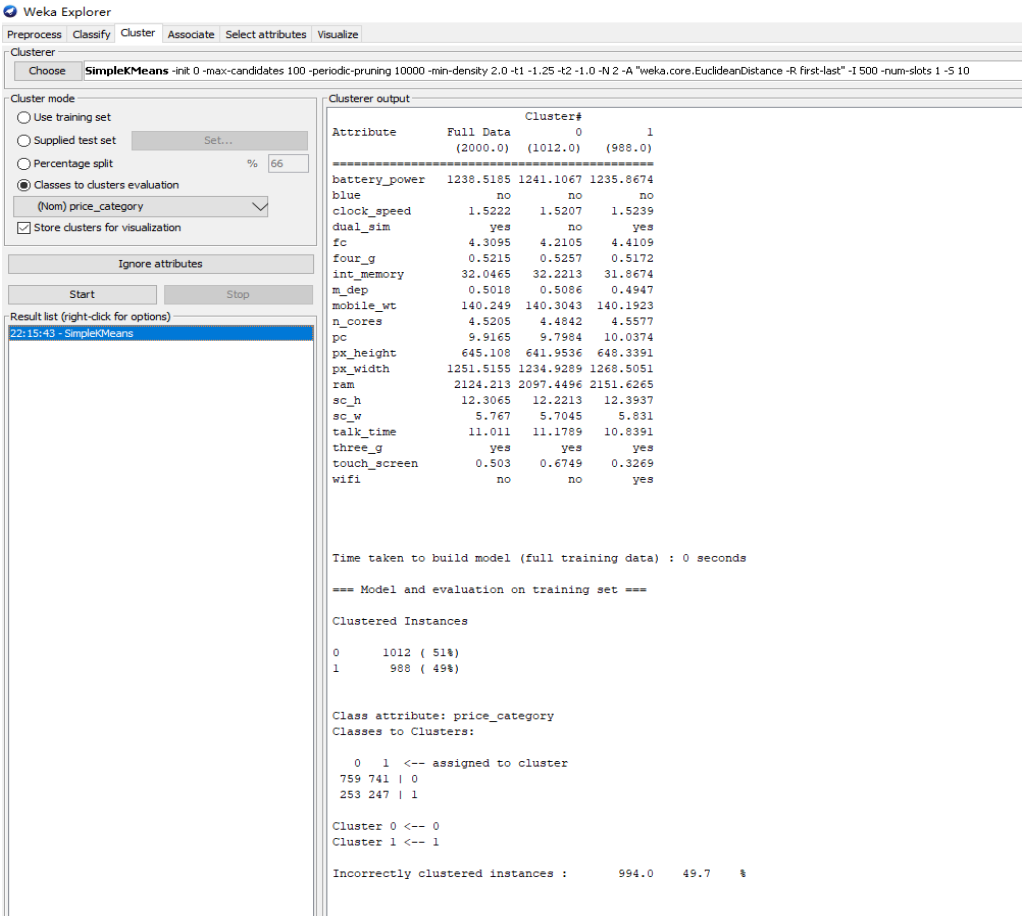


Figure 39 K-means results using original data

We noticed that the 20 attributes have 6 binary attributes: blue, dual_sim, four_g, three_g, touch_screen, wifi. So, we convert these attributes into binary (0 or 1) form. And then perform discretization on the attributes which have too many distinct values. This step can reduce the influence of extreme value on the result to some extent.

In [5]:
```
df.loc[:,'battery_power'] = pd.cut(df.loc[:,'battery_power'],bins=[0,1000,1500,2000],labels=['Low','Medium','High'])
df.loc[:,'px_height'] = pd.cut(df.loc[:,'px_height'],bins=[0,653,1307,2000],labels=['Low','Medium','High'])
df.loc[:,'px_width'] = pd.cut(df.loc[:,'px_width'],bins=[0,1000,1500,2000],labels=['Low','Medium','High'])
df.loc[:,'ram'] = pd.cut(df.loc[:,'ram'],bins=[0,1500,2750,4000],labels=['Low','Medium','High'])
df.loc[:,'mobile_wt'] = pd.cut(df.loc[:,'mobile_wt'],bins=[0,120,160,300],labels=['Low','Medium','High'])
df.loc[:,'int_memory'] = pd.cut(df.loc[:,'int_memory'],bins=[0,22.67,43.33,70],labels=['Low','Medium','High'])
df.loc[:,'clock_speed'] = pd.cut(df.loc[:,'clock_speed'],bins=[0,1.3,2.17,3],labels=['Low','Medium','High'])
```

In [6]:
```
df.head()
```

Out[6]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	px_height	px_width	ram	sc_h	sc_w	talk_time	thr
0	801	Medium	1	Low	1	15	0	Low	0.1	Low	...	Low	Low	Low	19	1	3	
1	1536	Medium	1	Low	1	9	0	High	0.2	Low	...	Low	Low	Medium	15	10	18	
2	1314	High	0	Low	0	5	1	Medium	0.6	Medium	...	Low	High	Low	5	1	12	
3	1963	High	1	Medium	1	3	0	High	0.2	Low	...	Low	Low	Medium	13	5	4	
4	1878	Low	1	Low	1	0	0	High	0.9	Low	...	Low	Medium	Medium	16	8	7	

5 rows × 22 columns

Figure 40 Discretize numerical data into Low Medium High

The clustering result is shown as follows:

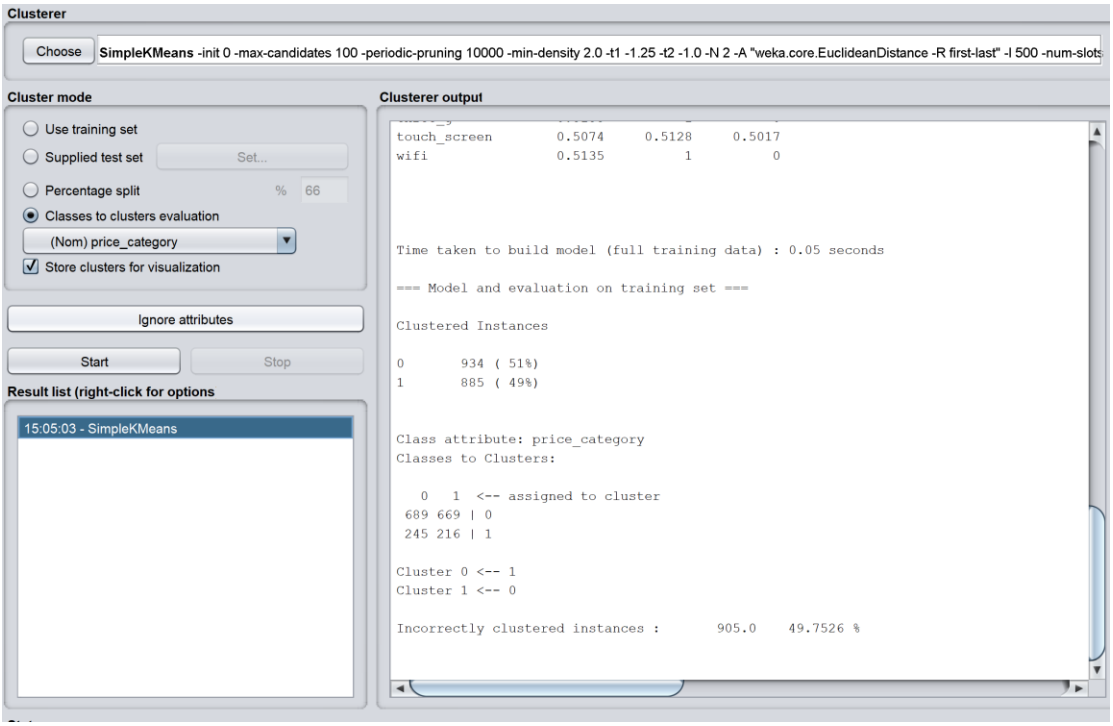


Figure 41 K-means results using discretized data

The result was not improved so much. It seems still like a random cluster, nearly 50%.

However, when the data is discretized into 3 bins: Low, Medium, High, some information was lost. When Weka is clustering, the distance

information is missing. Take n-cores as an example. If we discretize n_cores=[1,3] as Low, n_cores=[4,6] as Medium, n_cores=[7,9] as High, because the Low, Medium, High are nominal, they are at the same level. There will be a problem here. Obviously,

$$Distance(Low, High) > Distance(Low, Medium)$$

In Weka, when calculating the distance, $Distance(Low, High) = Distance(Low, Medium) = Distance(Medium, High)$, the relative relationship between them is lost.

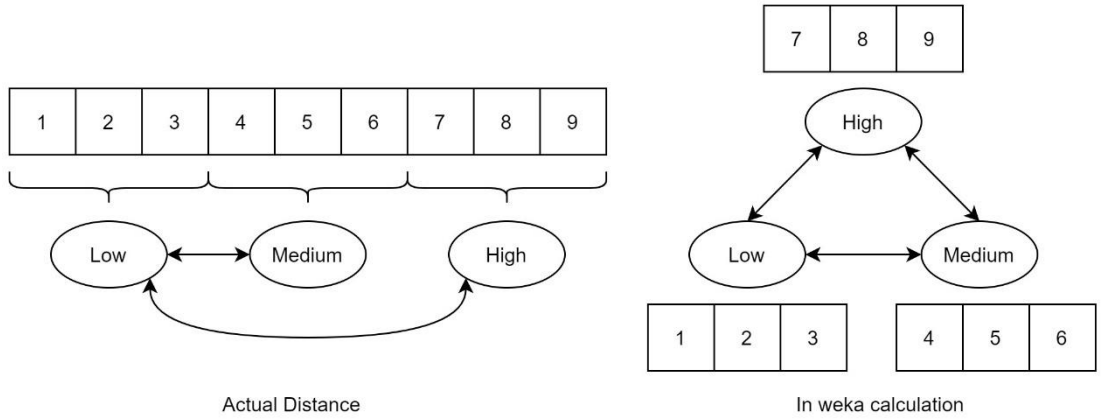


Figure 42 Distance information lost

A better way to deal with this problem is to use numerical values 0, 0.5 and 1 to replace Low, Medium and High, when discretizing.

```
In [59]:
df.loc[:,'battery_power'] = pd.cut(df.loc[:,'battery_power'],bins=[0,1000,1500,2000],labels=[0,0.5,1]).astype("float64")
df.loc[:,'px_height'] = pd.cut(df.loc[:,'px_height'],bins=[0,653,1307,2000],labels=[0,0.5,1]).astype("float64")
df.loc[:,'px_width'] = pd.cut(df.loc[:,'px_width'],bins=[0,1000,1500,2000],labels=[0,0.5,1]).astype("float64")
df.loc[:,'ram'] = pd.cut(df.loc[:,'ram'],bins=[0,1500,2750,4000],labels=[0,0.5,1]).astype("float64")
df.loc[:,'mobile_wt'] = pd.cut(df.loc[:,'mobile_wt'],bins=[0,120,160,300],labels=[0,0.5,1]).astype("float64")
df.loc[:,'int_memory'] = pd.cut(df.loc[:,'int_memory'],bins=[0,22.67,43.33,70],labels=[0,0.5,1]).astype("float64")
df.loc[:,'clock_speed'] = pd.cut(df.loc[:,'clock_speed'],bins=[0,1.3,2.17,3],labels=[0,0.5,1]).astype("float64")

In [60]:
df.head()

Out[60]:
```

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_
0	801	0.5	1	0.0	1	15	0	0.0	0.1	0.0	...	0.0	0.0	0.0	19	1	3	
1	1536	0.5	1	0.0	1	9	0	1.0	0.2	0.0	...	0.0	0.0	0.5	15	10	18	
2	1314	1.0	0	0.0	0	5	1	0.5	0.6	0.5	...	0.0	1.0	0.0	5	1	12	
3	1963	1.0	1	0.5	1	3	0	1.0	0.2	0.0	...	0.0	0.0	0.5	13	5	4	
4	1878	0.0	1	0.0	1	0	0	1.0	0.9	0.0	...	0.0	0.5	0.5	16	8	7	

5 rows × 22 columns

Figure 43 Discretize numerical data into 0, 0.5, 1

The result is shown as below:

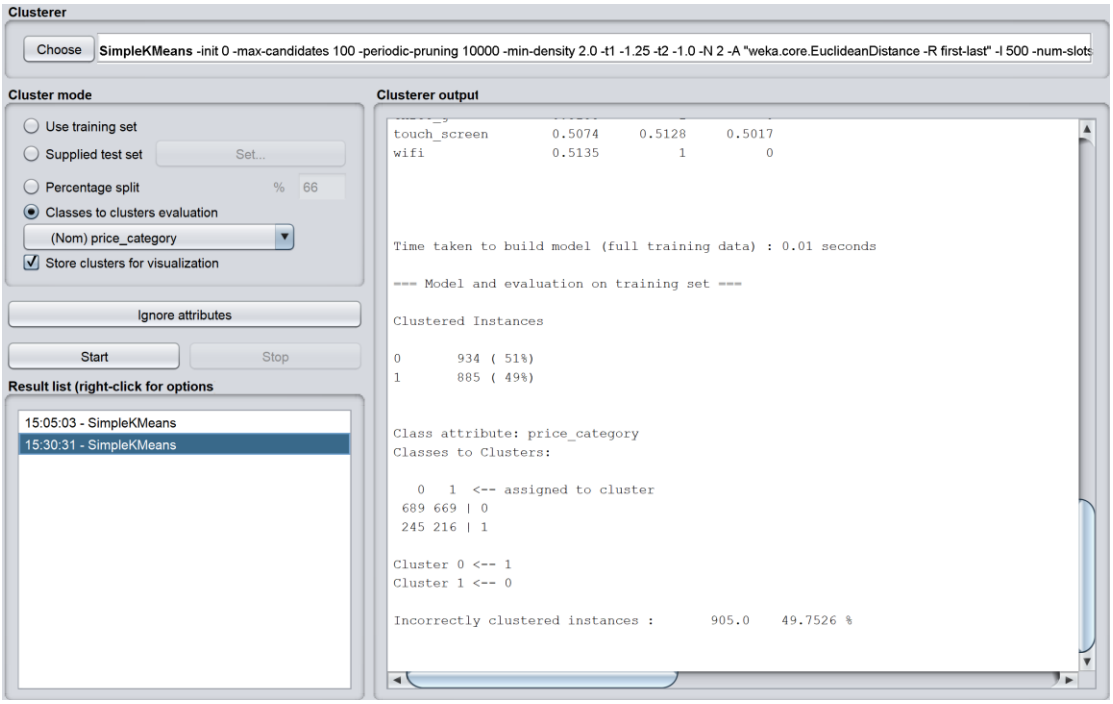


Figure 44 K-means results using discretized data

However, the result is still bad.

Then we use the same data to do Density Based clustering.

```
Class attribute: price_category
Classes to Clusters:

   0   1  <-- assigned to cluster
 761 739 | 0
 253 247 | 1

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

Incorrectly clustered instances :       992.0     49.6     %
```

Figure 45 DB cluster using original data

```
Class attribute: price_category
Classes to Clusters:

   0   1  <-- assigned to cluster
 689 669 | 0
 245 216 | 1

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

Incorrectly clustered instances :       905.0     49.7526 %
```

Figure 46 DB cluster using Low Medium High discretization

```
Class attribute: price_category
Classes to Clusters:

   0   1  <-- assigned to cluster
 689 669 | 0
 245 216 | 1

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

Incorrectly clustered instances :       905.0    49.7526 %
```

Figure 47 DB cluster using 0 0.5 1 discretization

5.3 Analysis

From the results, both the K-means cluster and Density Based cluster did not get good results. The incorrectly clustered instances take nearly 50%, which is like a coin toss problem.

To explore the reason, we need to focus on the procedure of clustering. About K-means clustering, the first step is to select initial clustering centres. The data points are distributed in attribute space. However, the class to cluster evaluation (price_category) is also an attribute. The same as Density Based clustering, the first step is to select a point randomly. The distribution of data points is not distributed by price_category, and the clustering is also not clustered according to price. The clustering result maybe fit the price_category, maybe not. It is an unsupervised procedure, not supervised. When we use price_category to evaluate the result, the result is also randomly. We tried many methods, but the result of clustering is not good. This shows that the data distribution is not highly correlated with the price of mobile phones. That is the reason why the result is bad all the time. One the other hand, the clustering results are poor at predicting mobile phone prices. That is because the data distribution of the processed data set is not highly correlated with the price of mobile phones.

6 Task 5 - Data Reduction

Attribute Reduction

In previous sections, we have performed some data reduction, such as reduce the number of attributes by combining multiple related attributes into a new attribute by Mathematical. In the data processor for the decision tree model, we converted the data of px_height and px_width into one new attribute data px_size by calculating the diagonal. We generated a new attribute PPI to replace px_height, px_width, sc_h and sc_w to reduce the number of attributes. Besides, we also implemented other attribute reduction methods, such as using information gain ratio rank to reduce attributes that have an insignificant impact on the target class. For example, in task 2, we used the GainRatioAttributeEval algorithm to remove 11 fewer impact attributes from the data set.

In this section, we will implement PCA for data reduction. Use the numerical data sheet created before. In this data sheet, all binary variables are in 0, 1 form instead of "Yes" or "No".

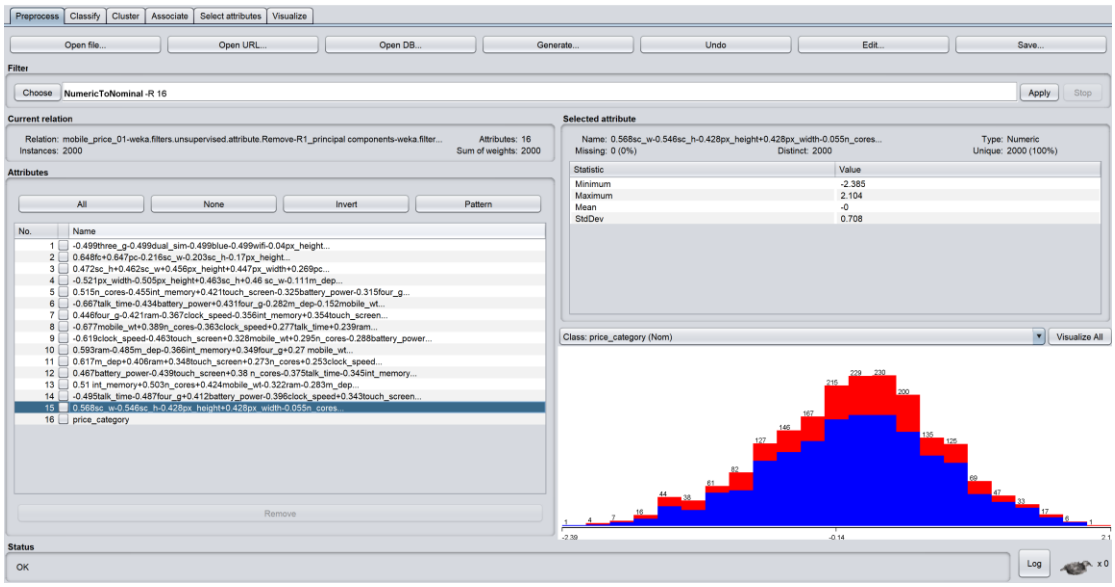


Figure 48 Do PCA using Weka

Then save the PCA data file as mobile_price_pca.arff.

Train the SVM, decision tree, and Naïve Bayes on the PCA data, and we get the following results:

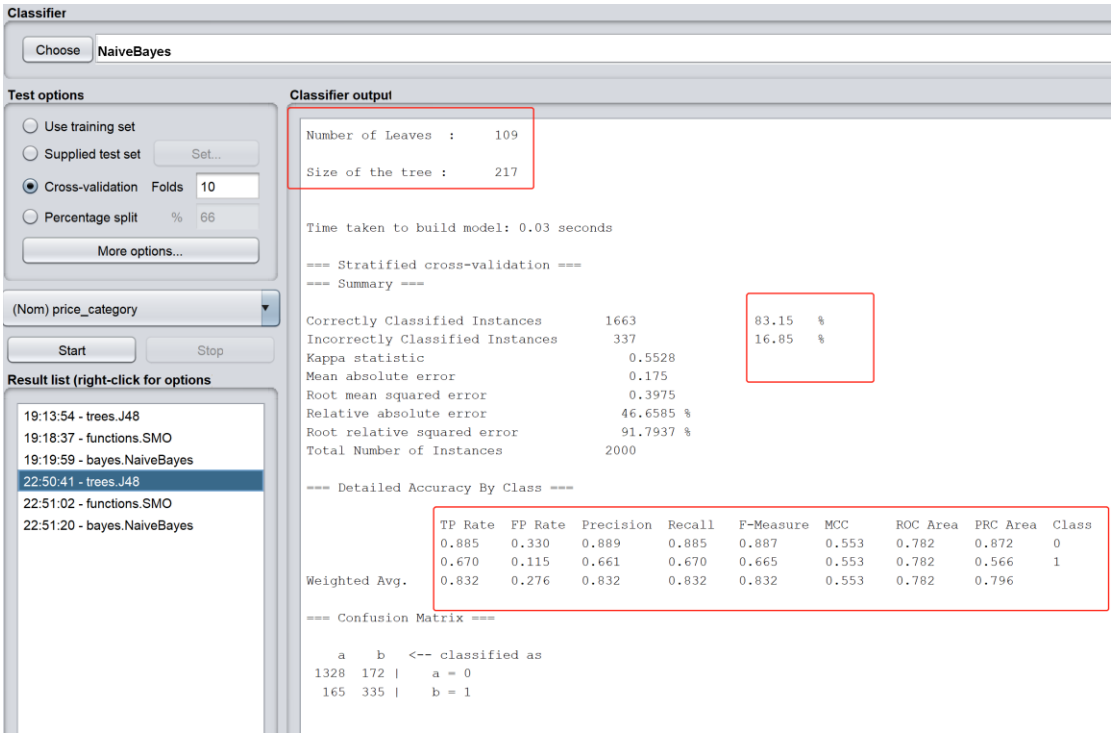


Figure 49 Result of decision tree on PCA data

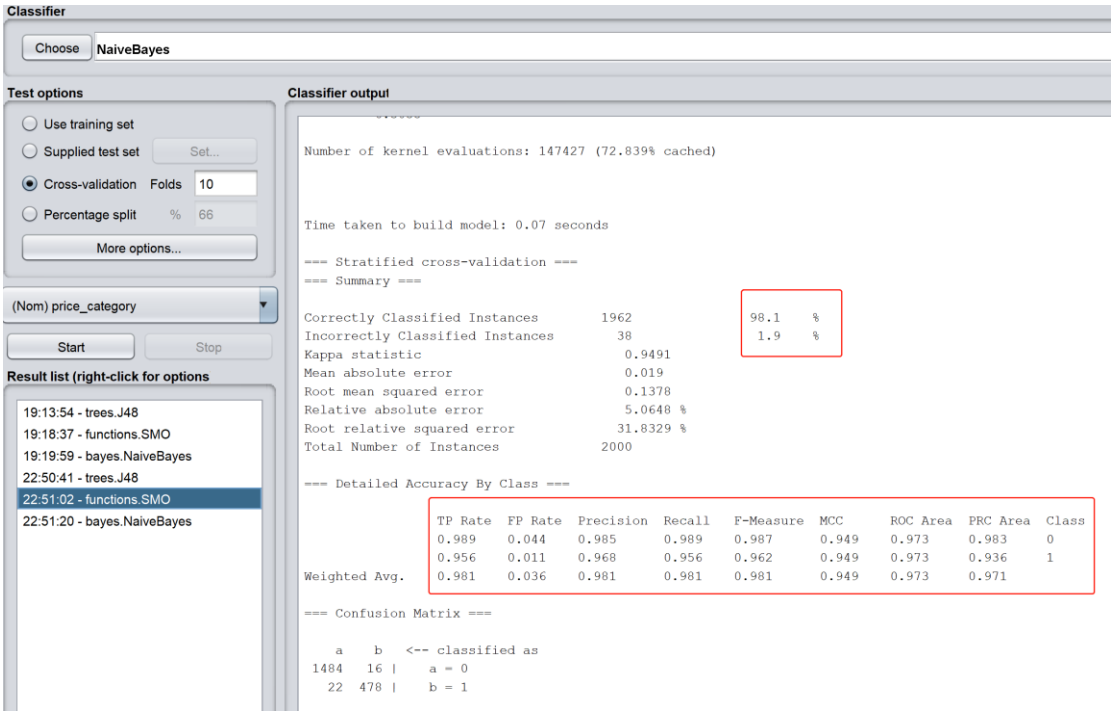


Figure 50 Result of SVM classifier on PCA data

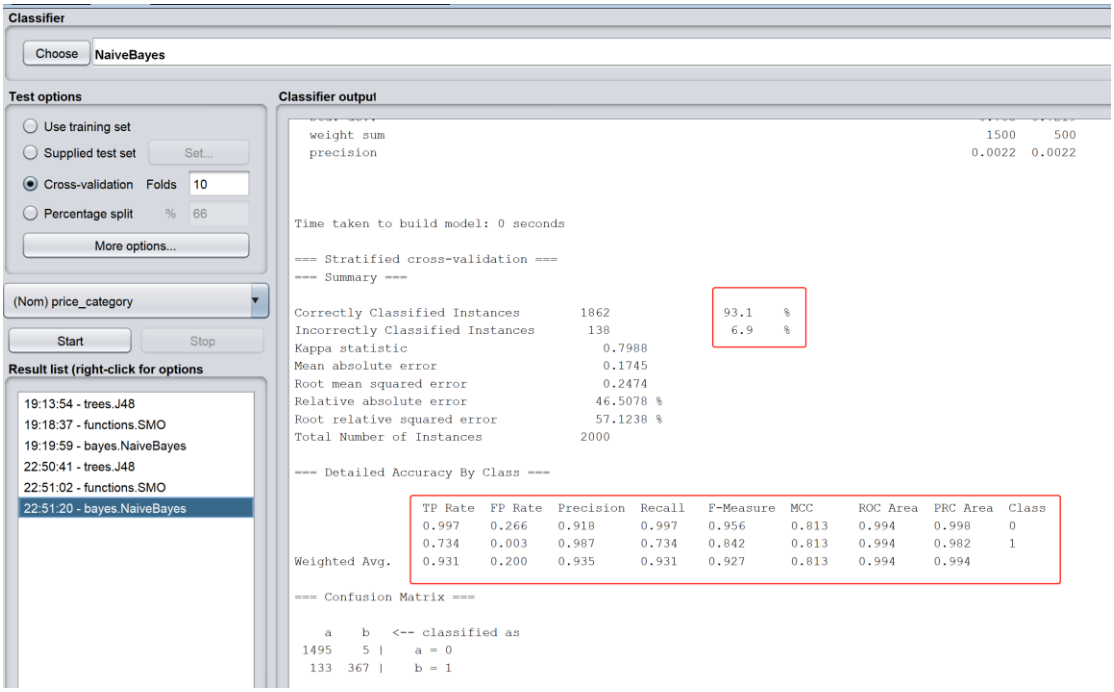


Figure 51 Result of Naïve Bayes classifier on PCA data

By comparing the result of PCA with that obtained in Section3, we found that PCA's evaluation decreased. For the decision tree, the number of leaves and the size of the tree are increased dramatically. The precision of the high price_category class is decreased by 31.5%, and recall is decreased by 31.2%. The overall prediction accuracy of the decision tree is decreased by 12%. PCA data performs bad on the decision tree. For SVM classifier, the precision of the high price_category class is increased by 1.5%, and recall is decreased by 1.2%. The overall prediction accuracy of SVM is increased by 0.1%. The result of SVM did not change a lot. For Naïve Bayes classifier, the precision of high price_category class is increased by 19.8%, and recall is decreased by 14.6%. The overall prediction accuracy of Naïve Bayes classifier is increased by 3.1%.

The final data dimensions obtained by PCA are calculated from the original dimensions. So the dimensions of PCA are related to each other, but not completely independent. However, Naive Bayes requires that the variables be independent of each other. This leads to a decline in the effectiveness of Naive Bayes.

One the other hand, the meaning of each feature dimension of the principal component is fuzzy to a certain extent, which is not as explanatory as the feature of the original sample. Non-principal components with small contribution may also contain important information for sample

differences, because dimension reduction may have an impact on subsequent data processing[24]. PCA, as a dimension reduction method, is used to deal with high-dimensional data, rather than to improve the accuracy of the model. In the SVM model training, the time loss of PCA data (0.07s) is far less than the original data (0.19s). PCA reduces the dimensions of the data while preserving the characteristics of the original data as much as possible. It is not lossless to the data itself. To some extent, although PCA greatly reduces the model operation time, it may have a negative impact on the model effect.

Numerosity Reduction

Numerosity reduction refers to reducing data volume by choosing alternative, smaller forms of data representation. In this project, we use sampling to reduce data volume. We use stratified sampling to avoid oversampling. There are 1358 low-price mobile phone data and 461 high-price mobile phone data, so we randomly choose 679 low-price data and 230 high-price data using python script. Then we use the sampling data to train classifiers in Task 3.

We trained decision tree, SVM, and Naïve Bayes on 3 different random sampling data sets. The results are shown in Appendix.

We compared the overall correct classification rate and recall. From the result, we got:

Table 1 High price mobile phone oveall comparison

Classifier	Decision Tree	SVM	Naïve Bayes
Sampling data1	-0.027	-0.016	-0.007
Sampling data2	-0.018	-0.010	-0.003
Sampling data3	-0.028	-0.008	-0.006

Table 2 High price mobile phone recall comparison

Classifier	Decision Tree	SVM	Naïve Bayes
Sampling data1	-0.054	-0.068	-0.023
Sampling data2	-0.036	-0.058	-0.048
Sampling data3	-0.053	-0.061	+0.015

We found a downward trend in all performance. In sampling data 3, only the recall of high price mobile phone increased 0.015.

Q: Does data reduction improve the quality of the classifiers?

According to the performance of the three classifiers on mobile price data, the answer is No. As the data set is reduced, less information is contained compared to the original data. Therefore, the effect of the model is reduced. As for the improvement of Naïve Bayes performance on sampling data 3, it should be a coincidence.

7 Task 6 - Comparison of three types of learning

This section will discuss three types of learning we used in this project: association rule mining, classification, and clustering.

In the project, we need to figure out which properties affect the price_category attribute based on the data provided by the project. The data is mixed data, containing both number attributes and category attributes, and has a high dimension with 22 different attributes. There is no missing value in the data set, but there may be outliers. For example, the screen length and pixel width both have 0 values, so the screen length-width ratio would appear very unreasonable, but these outliers can be understood as 0 values that exist after being processed somehow. These factors may cause outlier.

Association Rule Mining

Association rule mining is often used to find a relationship between the different set of items. It usually calls the role in the context of the purposes of marketing strategy, product design (like how to design a high price phone), and business decision making [25]. Traditionally, the association

rules are used to find business trends by analyzing consumer transactions [26].

Association rules extract valuable information from a large amount of data and can be understood as descriptions of data sets. In our project, the Apriori algorithm is used to mine the hidden association between data. So that the attribute that will have a high influence on the high price label can be found out through the rule. For example, in the results of association rule mining in WEKA, $rule\ 810.ram =' (2711.5 - \max)'\ 666 \Rightarrow price_{category} = High\ 473\ conf: (0.71)$ gives a good example.

This rule means that there are 666 phones in the data set with RAM greater than 2711.5, among which 473 phones have a high price label. The $lift = 2.84 > 1$, so $ram > 2711$ and price_category=high are positive occurrences. The higher the $lift$, the more significant the effect. In this way, we can easily find the attributes that expensive phones need to have. It should be noted that association rules are a kind of information or knowledge mining, which can only show the association between data. It does not mean that machine learning is not capable of prediction. The results of association rule mining will not be affected by data noise; however, the data must be discrete.

Association rules mining also have some limitations. The results of association mining may contain missing leading rules. Association rule mining finds correlation between attributes, not causality. For example, there is a rule "$hospitalized = yes\ 100 \Rightarrow death\ =\ yes\ 10$". We can only detect by this rule that 100 people are hospitalized, 10 of them die during the hospitalization. It does not mean that a person died because they were hospitalized.

In addition, when item set A contains attributes, if there is an attribute whose lift is very high, the result will also be disturbed. In our task 2 rule $No.784\ ram = '(2712 - max)'\ wifi = No\ 313\ \Rightarrow price\_category = High\ 225\ \ conf: (0.72)$. It can be a high-priced phone even without $WiFi$. This is because the $lift$ of ram is so high than others, and it fades the other attributes. As a result, no matter which it pairs with, the result will be a high-price phone. Therefore, we need the experience to determine whether the rule is true or not. This is also the reason why our rule selection is not strictly based on the Metric.

Classification

Classification is to divide data into different categories according to their attributes. It can map data items in the database to a given category. By analyzing the observation results, we can observe the internal relations and differences between the data and provide an important basis for further data. There are many classification methods, decision tree, SVM, Bayes classifier, etc. The classification process is divided into two stages: learning and testing[27]. During the learning stage, some known data will be input to build the model. In a later testing stage, the established model is tested by using most of the other data to determine the accuracy of the model. In this step, cross-validation is often used to evaluate the predictive effectiveness of the model. If the accuracy is limited, the model can be used to predict unknown data classes.

One obvious difference between classification and association rule mining is that classification can do prediction. We trained three different classifiers, decision tree, SVM, and Naïve Bayes classifier. Decision trees are not sensitive to outlier and are more suitable for processing samples with missing attribute values. It has both predictive and mining capabilities, and it is easy to explain[28]. SVM is good at dealing with numerical variables and high-dimensional data. However, SVM is highly sensitive to data noise. And when the training data set is very large, the model training will be time-consuming [29]. Naive Bayes can process multi-classification problems. It has a good ability of prediction and mining. But it is sensitive to the representation of the input data[30]. Naive Bayes requires that all variables be independent, and the numerical data to be normal distribution. Therefore, we must process our training data set in the project to reduce the error by discretization.

Clustering

Clustering is also to divide data into different categories. But clustering is different from classification, it is an unsupervised learning method [20]. Clustering is used to identify natural groups based on an attribute group, the data in each group has similar attribute features. So clustering is based on the data without specific data classes (target attribute). In the clustering procedure, every attribute has the same importance. If other attributes are added, it will affect the result.

We used K-means and Density Based Clustering in the project. Both clustering methods extract the similar attribute features and form a class,

without reference to price_category. In other words, there are some attributes that have very little correlation to price_category in the mobile price data set. These attributes should be weighted down in the process of clustering, but their importance is not diminished so that we got bad clustering results. That is the reason why the clustering result is bad evaluated by price_category in the mobile phone price data set.

Reference

[1]. CITS5504 Data Warehousing SEM 1 2021, Project 2 description on LMS

[2]. Frawley, W. J., Piatetsky-Shapiro, G., & Matheus, C. J. (1992). Knowledge discovery in databases: An overview. AI magazine.

[3]. Zeyi W. CITS5504 Data Warehousing SEM 1 2021, Lecture slides 8 Frequent Patterns and Association Rule Mining, p3-p5.

[4]. Zeyi W. CITS5504 Data Warehousing SEM 1 2021, Lecture slides 8 Frequent Patterns and Association Rule Mining, p36.

[5]. Decision tree - Wikipedia. (2021). Retrieved 20 May 2021, from https://en.wikipedia.org/wiki/Decision_tree

[6]. Molnar, C. (2021). 4.5 Decision Rules | Interpretable Machine Learning. Retrieved 20 May 2021, from https://christophm.github.io/interpretable-ml-book/rules.html

[7]. Chauhan, N. (2021). Decision Tree Algorithm, Explained - KDnuggets. Retrieved 20 May 2021, from https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

[8]. Cortes, C.; Vapnik, V. Support-vector networks. Machine Learning. 1995, 20 (3): 273–297. doi:10.1007/BF00994018.

[9]. Zeyi W. CITS5504 Data Warehousing SEM 1 2021, Lecture slides 9 Pattern-based Classification and Supervised Learning

[10]. SVM in R for Data Classification using e1071 Package - TechVidvan. (2020). Retrieved 20 May 2021, from https://techvidvan.com/tutorials/svm-in-r/

[11]. McCallum, A. "Graphical Models, Lecture2: Bayesian Network Represention"

[12]. Priyam, Anuja, et al. "Comparative analysis of decision tree classification algorithms." International Journal of current engineering and technology 3.2 (2013): 334-337.

[13]. "Display size - Wikipedia", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Display_size. [Accessed: 20- May- 2021].

[14]. “Display resolution - Wikipedia", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Display_resolution. [Accessed: 20- May- 2021].

[15]. Chauhan H., "Implementation of decision tree algorithm c4.5", 2013. [Accessed 20 May 2021].

[16]. Joachims, T. (1998). Making large-scale SVM learning practical (No. 1998, 28). Technical report.

[17]. Boullé M., "MODL: A Bayes optimal discretization method for continuous attributes", Machine Learning, vol. 65, no. 1, pp. 131-165, 2006. Available: 10.1007/s10994-006-8364-x

[18]. "Kendall rank correlation coefficient - Wikipedia", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Kendall_rank_correlation_coefficient. [Accessed: 20- May- 2021].

[19]. Matsuzaki T., "Mathematical Introduction for SVM and Kernel Functions", tsmatz, 2021. [Online]. Available: https://tsmatz.wordpress.com/2020/06/01/svm-and-kernel-functions-mathematics/. [Accessed: 20- May- 2021].

[20]. Silwattananusarn T., KulthidaTuamsuk A. D., "Data Mining and Its Applications for Knowledge Management : A Literature Review from 2007 to 2012," International Journal of Data Mining & Knowledge Management Process, vol. 2, no. 5, 2012.

[21]. Inference.Org.Uk, 2021, http://www.inference.org.uk/mackay/itprnn/ps/284.292.pdf. Accessed 17 May 2021.

[22]. Geoffrey I. et al. "Density-Based Clustering". Encyclopedia Of Machine Learning, 2011, pp. 270-273. Springer US, doi:10.1007/978-0-387-30164-8_211. Accessed 19 May 2021.

[23]. "DBSCAN - Wikipedia". En.Wikipedia.Org, 2021, https://en.wikipedia.org/wiki/DBSCAN. Accessed 19 May 2021.

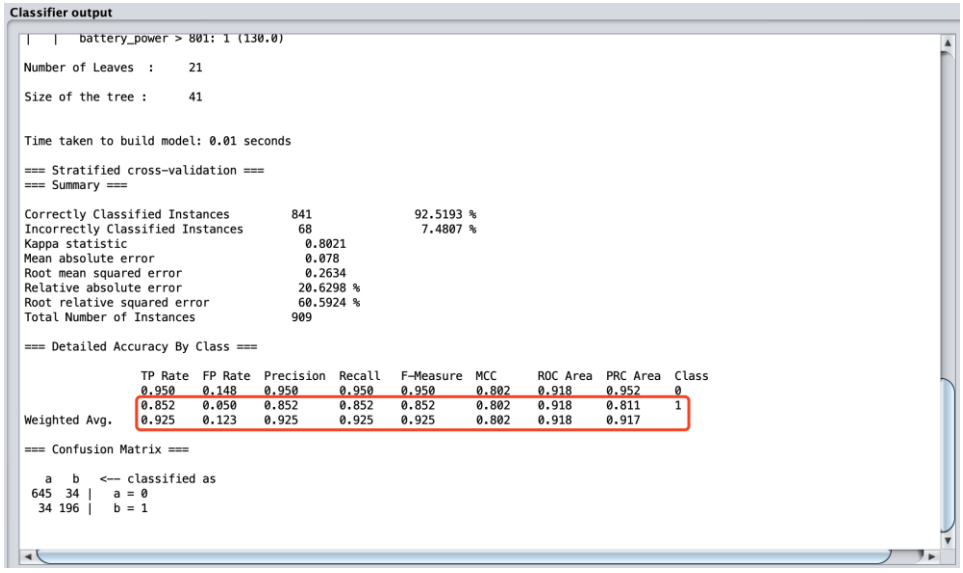
[24]. Abbott, D. (2014). Applied predictive analytics: Principles and techniques for the professional data analyst. Wiley. Jiang, H., & Eskridge, K. M. (2000). BIAS IN PRINCIPAL COMPONENTS ANALYSIS DUE TO CORRELATED OBSERVATIONS. Conference on Applied Statistics in Agriculture. https://doi.org/10.4148/2475-7772.1247

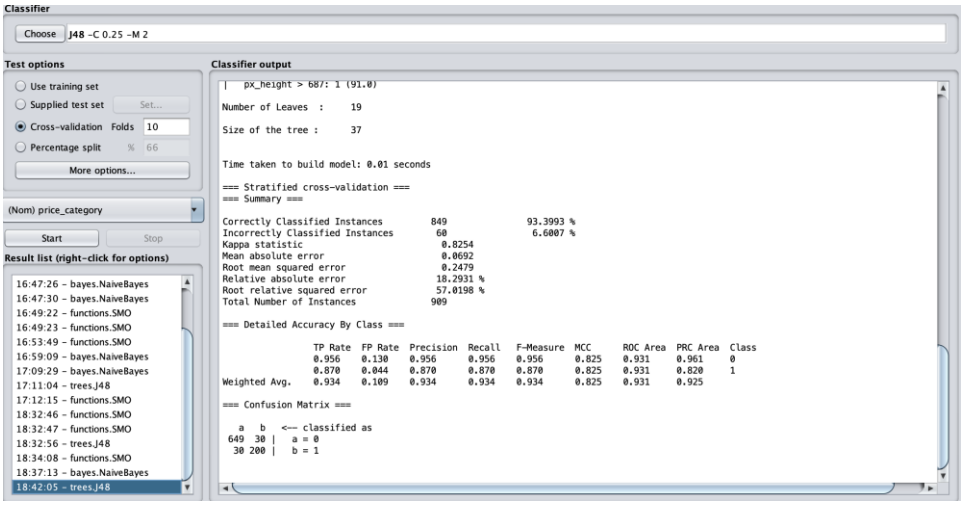
[25]. Marlina, L., Muslim, M., Siahaan, A. U., & Utama, P. (2016). Data Mining Classification Comparison (Naïve Bayes and C4. 5 Algorithms). Int. J. Eng. Trends Technol, 38(7), 380-383.
[26]. Rajagopal S., "Customer Data Clustering Using Data Mining Technique," International Journal of Database Management Systems, vol. 3, no. 4, pp. 1-11, 2011.
[27]. Fitriani W. and Siahaan A. P. U., "Comparison Between WEKA and Salford Systemin Data Mining Software," International Journal of Mobile Computing and Application, vol. 3, no. 4, pp. 1-4, 2016.
[28]. Priyam, Anuja, et al. "Comparative analysis of decision tree classification algorithms." International Journal of current engineering and technology 3.2 (2013): 334-337.
[29]. Zhijie L., et al. "Study on SVM compared with the other text classification methods." 2010 Second international workshop on education technology and computer science. Vol. 1. IEEE, 2010.
[30]. Bhardwaj, Brijesh Kumar, and Saurabh Pal. "Data Mining: A prediction for performance improvement using classification." arXiv preprint arXiv:1201.3418 (2012).

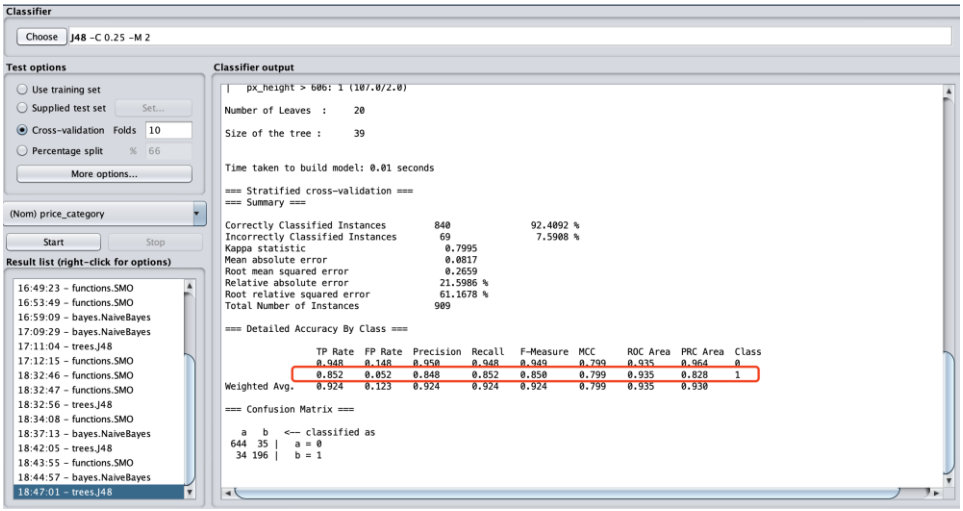
Appendix

*Result of classifiers on sampled data set

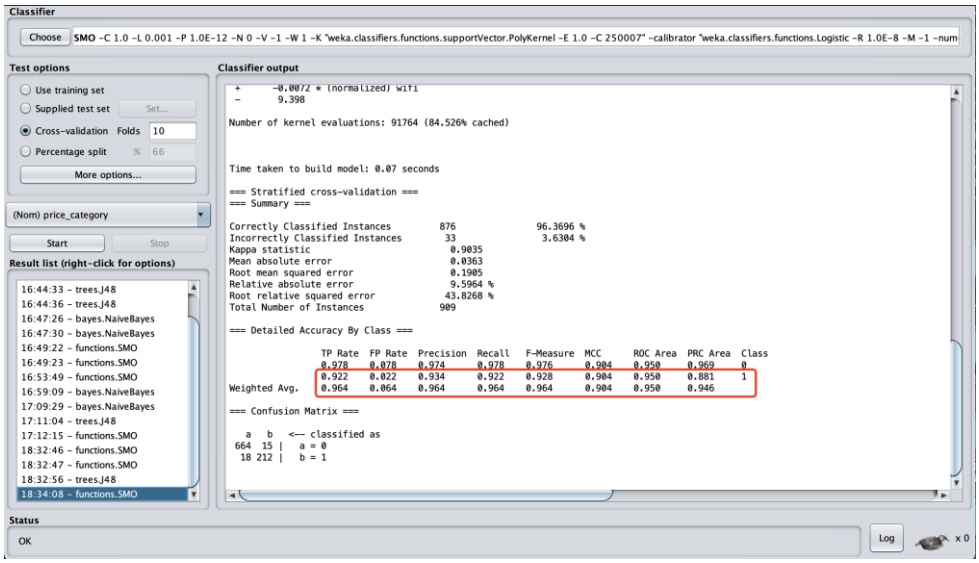
Decision Tree:

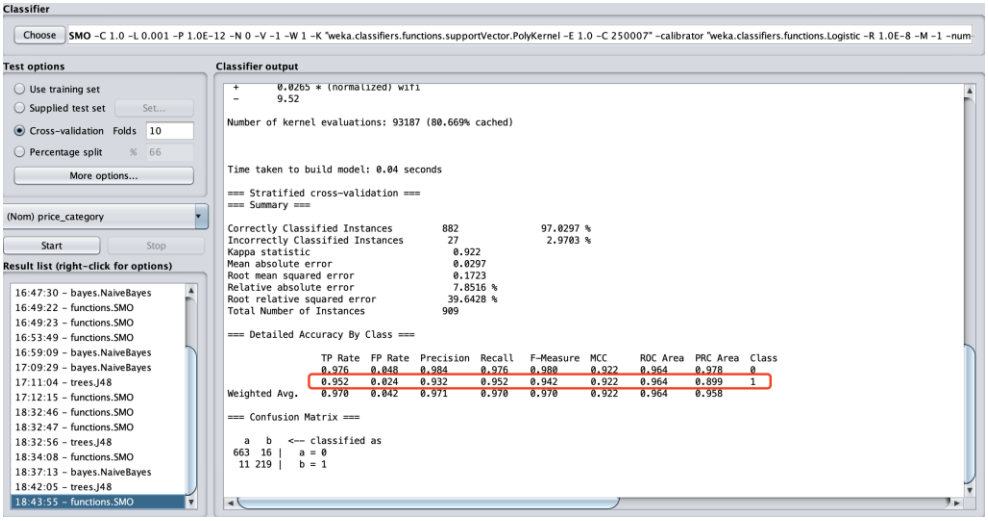


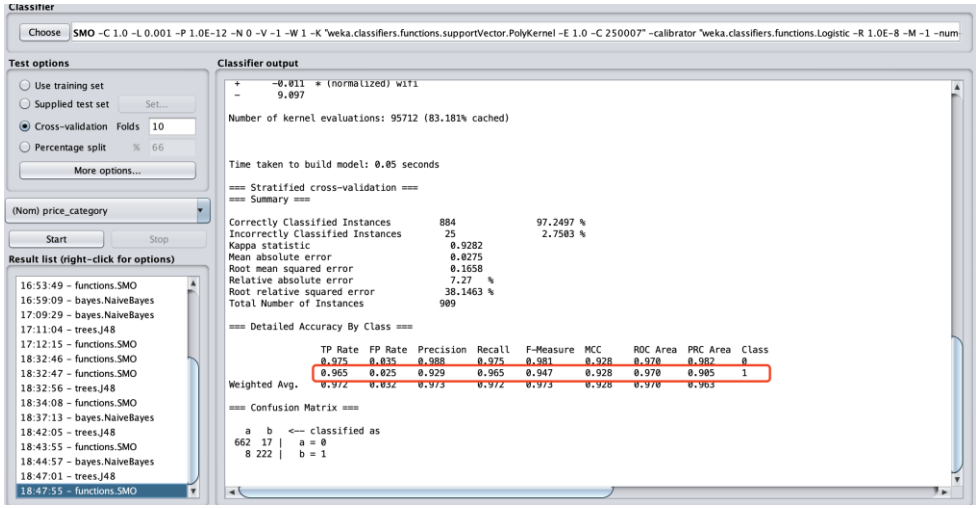




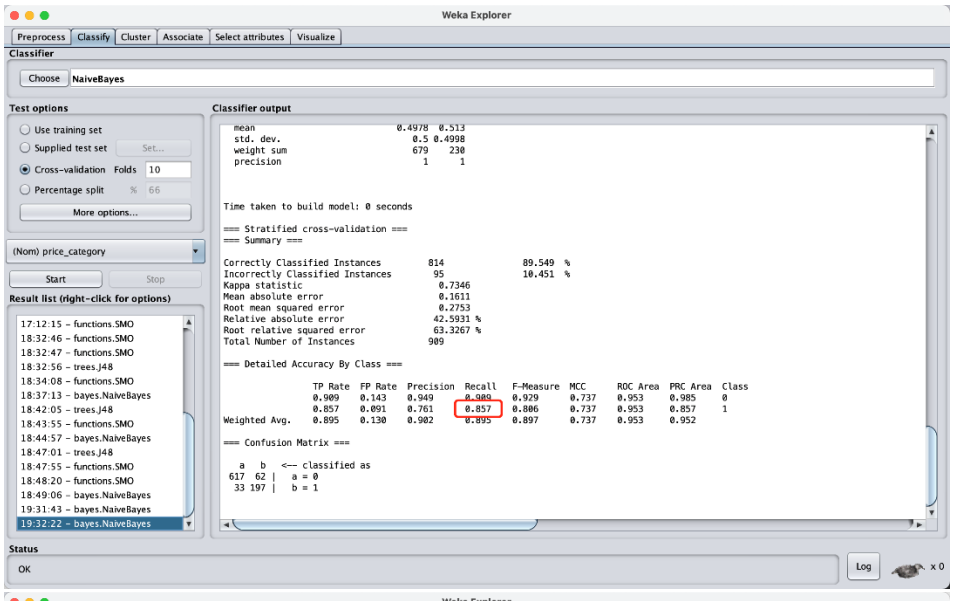
SVM:

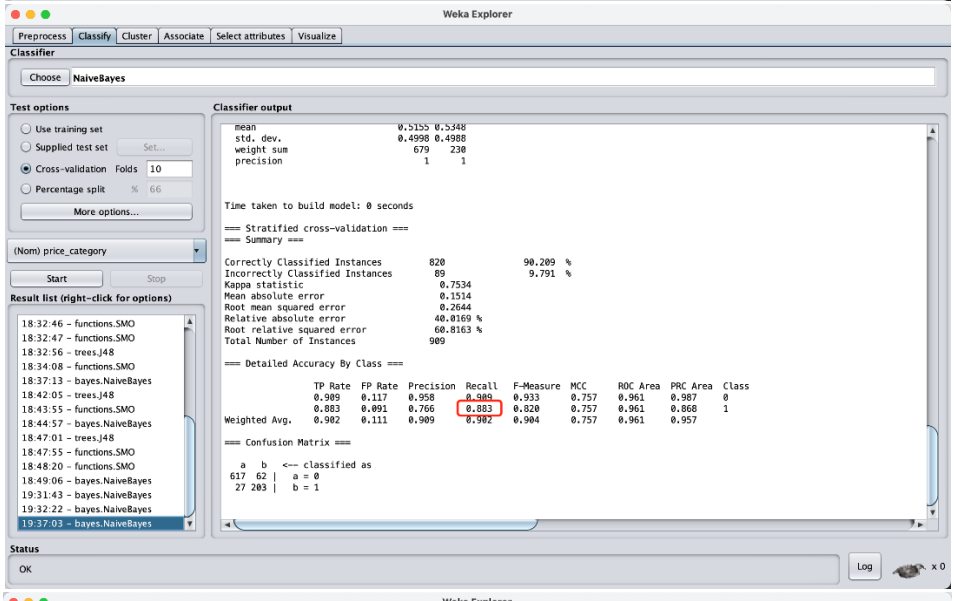

Classifier
Choose SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) price_category
Start Stop
Result list (right-click for options)
16:44:33 - trees.J48
16:44:36 - trees.J48
16:47:26 - bayes.NaiveBayes
16:47:30 - bayes.NaiveBayes
16:49:22 - functions.SMO
16:49:23 - functions.SMO
16:53:49 - functions.SMO
16:59:09 - bayes.NaiveBayes
17:09:29 - bayes.NaiveBayes
17:11:04 - trees.J48
17:12:15 - functions.SMO
18:32:46 - functions.SMO
18:32:47 - functions.SMO
18:32:56 - trees.J48
18:34:08 - functions.SMO
Classifier output
+ -0.0072 * (normalized) wifi
- 9.398
Number of kernel evaluations: 91764 (84.526% cached)
Time taken to build model: 0.07 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 876 96.3696 %
Incorrectly Classified Instances 33 3.6304 %
Kappa statistic 0.9035
Mean absolute error 0.0363
Root mean squared error 0.1905
Relative absolute error 9.5964 %
Root relative squared error 43.8268 %
Total Number of Instances 909
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.978 0.078 0.974 0.978 0.976 0.904 0.950 0.969 0
0.922 0.022 0.934 0.922 0.928 0.904 0.950 0.881 1
Weighted Avg. 0.964 0.064 0.964 0.964 0.964 0.904 0.950 0.946
=== Confusion Matrix ===
a b <-- classified as
664 15 | a = 0
18 212 | b = 1
Status
OK
Log
x 0

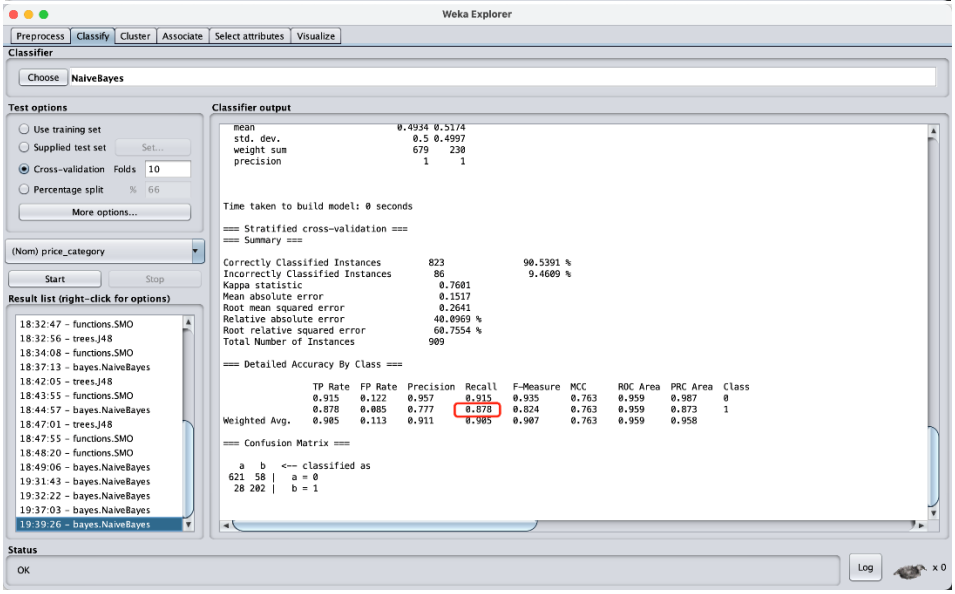

Classifier
Choose SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) price_category
Start Stop
Result list (right-click for options)
16:47:30 - bayes.NaiveBayes
16:49:22 - functions.SMO
16:49:23 - functions.SMO
16:53:49 - functions.SMO
16:59:09 - bayes.NaiveBayes
17:09:29 - bayes.NaiveBayes
17:11:04 - trees.J48
17:12:15 - functions.SMO
18:32:46 - functions.SMO
18:32:47 - functions.SMO
18:32:56 - trees.J48
18:34:08 - functions.SMO
18:37:13 - bayes.NaiveBayes
18:42:05 - trees.J48
18:43:55 - functions.SMO
Classifier output
+ 0.0265 * (normalized) wifi
- 9.52
Number of kernel evaluations: 93187 (80.669% cached)
Time taken to build model: 0.04 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 882 97.0297 %
Incorrectly Classified Instances 27 2.9703 %
Kappa statistic 0.922
Mean absolute error 0.0297
Root mean squared error 0.1723
Relative absolute error 7.8516 %
Root relative squared error 39.6428 %
Total Number of Instances 909
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.976 0.048 0.984 0.976 0.980 0.922 0.964 0.978 0
0.952 0.024 0.932 0.952 0.942 0.922 0.964 0.899 1
Weighted Avg. 0.970 0.042 0.971 0.970 0.970 0.922 0.964 0.958
=== Confusion Matrix ===
a b <-- classified as
663 16 | a = 0
11 219 | b = 1


Classifier
Choose SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) price_category
Start Stop
Result list (right-click for options)
16:53:49 - functions.SMO
16:59:09 - bayes.NaiveBayes
17:09:29 - bayes.NaiveBayes
17:11:04 - trees.J48
17:12:15 - functions.SMO
18:32:46 - functions.SMO
18:32:47 - functions.SMO
18:32:56 - trees.J48
18:34:08 - functions.SMO
18:37:13 - bayes.NaiveBayes
18:42:05 - trees.J48
18:43:55 - functions.SMO
18:44:57 - bayes.NaiveBayes
18:47:01 - trees.J48
18:47:55 - functions.SMO
Classifier output
+ -0.011 * (normalized) wifi
- 9.097
Number of kernel evaluations: 95712 (83.181% cached)
Time taken to build model: 0.05 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 884 97.2497 %
Incorrectly Classified Instances 25 2.7503 %
Kappa statistic 0.9282
Mean absolute error 0.0275
Root mean squared error 0.1658
Relative absolute error 7.27 %
Root relative squared error 38.1463 %
Total Number of Instances 909
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.975 0.035 0.988 0.975 0.981 0.928 0.970 0.982 0
0.965 0.025 0.929 0.965 0.947 0.928 0.970 0.905 1
Weighted Avg. 0.972 0.032 0.973 0.972 0.973 0.928 0.970 0.963
=== Confusion Matrix ===
a b <-- classified as
662 17 | a = 0
8 222 | b = 1

Naïve Bayes


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) price_category
Start
Stop
Result list (right-click for options)
17:12:15 - functions.SMO
18:32:46 - functions.SMO
18:32:47 - functions.SMO
18:32:56 - trees.J48
18:34:08 - functions.SMO
18:37:13 - bayes.NaiveBayes
18:42:05 - trees.J48
18:43:55 - functions.SMO
18:44:57 - bayes.NaiveBayes
18:47:01 - trees.J48
18:47:55 - functions.SMO
18:48:20 - functions.SMO
18:49:06 - bayes.NaiveBayes
19:31:43 - bayes.NaiveBayes
19:32:22 - bayes.NaiveBayes
Classifier output
mean 0.4978 0.513
std. dev. 0.5 0.4998
weight sum 679 230
precision 1 1
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 814 89.549 %
Incorrectly Classified Instances 95 10.451 %
Kappa statistic 0.7346
Mean absolute error 0.1611
Root mean squared error 0.2753
Relative absolute error 42.5931 %
Root relative squared error 63.3267 %
Total Number of Instances 909
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.909 0.143 0.949 0.909 0.929 0.737 0.953 0.985 0
0.857 0.091 0.761 0.857 0.806 0.737 0.953 0.857 1
Weighted Avg. 0.895 0.130 0.902 0.895 0.897 0.737 0.953 0.952
=== Confusion Matrix ===
a b <-- classified as
617 62 | a = 0
33 197 | b = 1
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) price_category
Start
Stop
Result list (right-click for options)
18:32:46 - functions.SMO
18:32:47 - functions.SMO
18:32:56 - trees.J48
18:34:08 - functions.SMO
18:37:13 - bayes.NaiveBayes
18:42:05 - trees.J48
18:43:55 - functions.SMO
18:44:57 - bayes.NaiveBayes
18:47:01 - trees.J48
18:47:55 - functions.SMO
18:48:20 - functions.SMO
18:49:06 - bayes.NaiveBayes
19:31:43 - bayes.NaiveBayes
19:32:22 - bayes.NaiveBayes
19:37:03 - bayes.NaiveBayes
Classifier output
mean 0.5155 0.5348
std. dev. 0.4998 0.4988
weight sum 679 230
precision 1 1
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 820 90.209 %
Incorrectly Classified Instances 89 9.791 %
Kappa statistic 0.7534
Mean absolute error 0.1514
Root mean squared error 0.2644
Relative absolute error 40.0169 %
Root relative squared error 60.8163 %
Total Number of Instances 909
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.909 0.117 0.958 0.909 0.933 0.757 0.961 0.987 0
0.883 0.091 0.766 0.883 0.820 0.757 0.961 0.868 1
Weighted Avg. 0.902 0.111 0.909 0.902 0.904 0.757 0.961 0.957
=== Confusion Matrix ===
a b <-- classified as
617 62 | a = 0
27 203 | b = 1
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) price_category
Start
Stop
Result list (right-click for options)
18:32:47 - functions.SMO
18:32:56 - trees.J48
18:34:08 - functions.SMO
18:37:13 - bayes.NaiveBayes
18:42:05 - trees.J48
18:43:55 - functions.SMO
18:44:57 - bayes.NaiveBayes
18:47:01 - trees.J48
18:47:55 - functions.SMO
18:48:20 - functions.SMO
18:49:06 - bayes.NaiveBayes
19:31:43 - bayes.NaiveBayes
19:32:22 - bayes.NaiveBayes
19:37:03 - bayes.NaiveBayes
19:39:26 - bayes.NaiveBayes
Classifier output
mean 0.4934 0.5174
std. dev. 0.5 0.4997
weight sum 679 230
precision 1 1
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 823 90.5391 %
Incorrectly Classified Instances 86 9.4609 %
Kappa statistic 0.7601
Mean absolute error 0.1517
Root mean squared error 0.2641
Relative absolute error 40.0969 %
Root relative squared error 60.7554 %
Total Number of Instances 909
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.915 0.122 0.957 0.915 0.935 0.763 0.959 0.987 0
0.878 0.085 0.777 0.878 0.824 0.763 0.959 0.873 1
Weighted Avg. 0.905 0.113 0.911 0.905 0.907 0.763 0.959 0.958
=== Confusion Matrix ===
a b <-- classified as
621 58 | a = 0
28 202 | b = 1
Status
OK
Log
x 0