Data Warehousing Project 2 – 22704832, 22705507

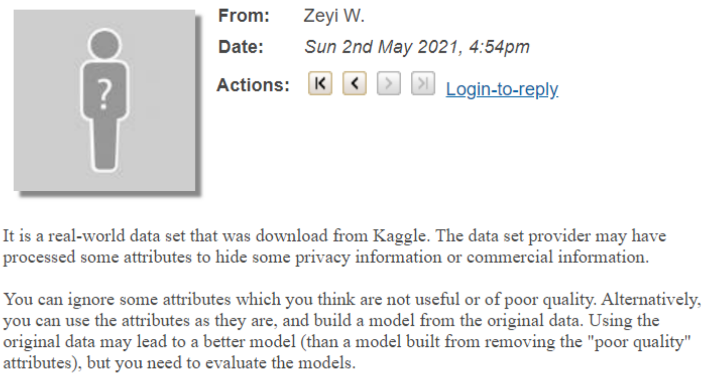
CITS3401 Data Warehousing Project 2

Jake Lyell (2704832), Jordan Lee (22705507)

This report will detail the process the students Jake Lyell (22704832) and Jordan Lee (22705507) took to process the given data set. This report contains answers to the six listed tasks as well as 'or screenshots (for Excel, or other data processing software) of data cleaning and process procedures.' The 'Intermediate and final result files for all data processing procedures.' I.e., the WEKA save files are included in the submitted ZIP folder.

1. Data Cleaning and Analysis

The first step to take is to analyse the data set to determine which attributes will be cleaned or remove attributes. In the data set we concluded that the ID column needs to be removed. Which can be done through WEKA by using 'remove.' We have chosen to leave in the other attributes, as we will perform attribute reduction later. It can also be seen that the attribute fc has an outlier of many values of <1 megapixel. This is because there is a considerable number of phones with no front camera that has been recorded as 0. We decided to leave this attribute it in however because camera quality is an important metric for price and a consumer's desired qualities in phones. It should be noted that close examination of the data showed phones with unrealistic values such as 0 cm in screen width but some value in height (which is impossible). We have decided to leave these results in, in line with advice from the help forum



From: Zeyi W.

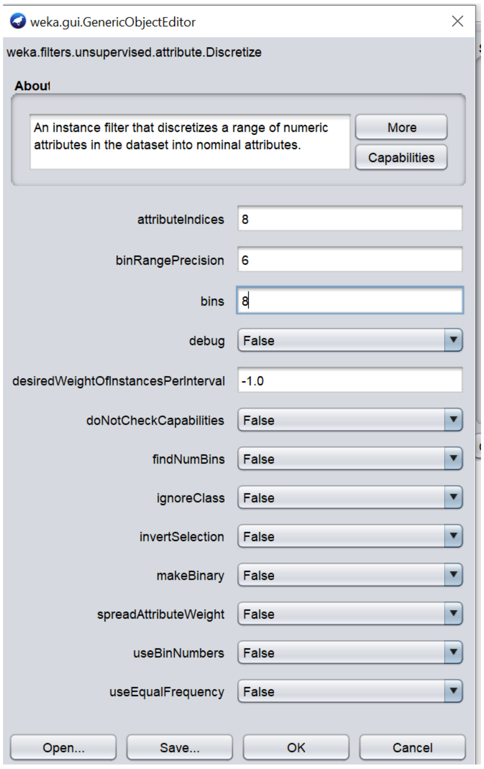
Date: *Sun 2nd May 2021, 4:54pm*

Actions:

It is a real-world data set that was download from Kaggle. The data set provider may have processed some attributes to hide some privacy information or commercial information.

You can ignore some attributes which you think are not useful or of poor quality. Alternatively, you can use the attributes as they are, and build a model from the original data. Using the original data may lead to a better model (than a model built from removing the "poor quality" attributes), but you need to evaluate the models.

Upon analysing the other attributes, we also determined that we will 'discretise' all the other numerical attributes. Continuous data can be defined as constant data values recorded over time. We concluded that the numerical attributes of the phones are not recorded over time. That is, a phone's attribute values remain stagnant and does not change. A phone does not transform from having say 4 cores to 8.



weka.gui.GenericObjectEditor

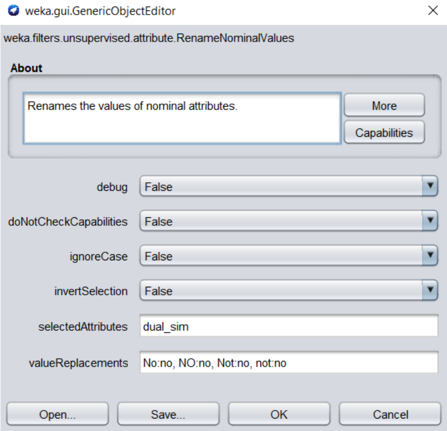
weka.filters.unsupervised.attribute.Discretize

About

An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes.

attributeIndices: 8
binRangePrecision: 6
bins: 8
debug: False
desiredWeightOfInstancesPerInterval: -1.0
doNotCheckCapabilities: False
findNumBins: False
ignoreClass: False
invertSelection: False
makeBinary: False
spreadAttributeWeight: False
useBinNumbers: False
useEqualFrequency: False

In this above step shown, this is discretising the n_cores attribute with 8 bins.



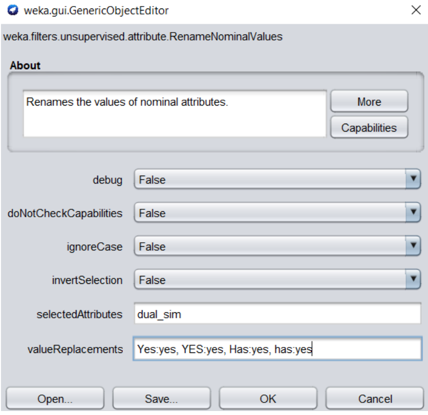
weka.gui.GenericObjectEditor

weka.filters.unsupervised.attribute.RenameNominalValues

About

Renames the values of nominal attributes.

debug: False
doNotCheckCapabilities: False
ignoreCase: False
invertSelection: False
selectedAttributes: dual_sim
valueReplacements: No:no, NO:no, Not:no, not:no



weka.gui.GenericObjectEditor

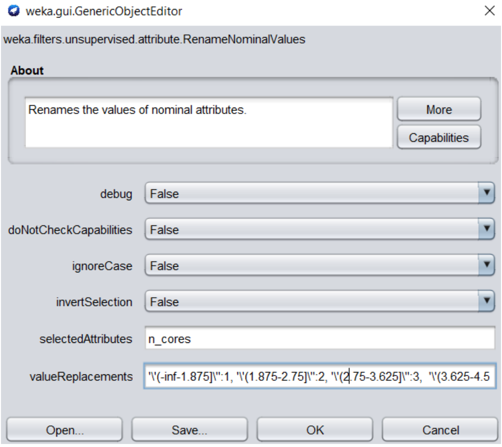
weka.filters.unsupervised.attribute.RenameNominalValues

About

Renames the values of nominal attributes.

debug: False
doNotCheckCapabilities: False
ignoreCase: False
invertSelection: False
selectedAttributes: dual_sim
valueReplacements: Yes:yes, YES:yes, Has:yes, has:yes

These above screenshots show use standardising the yes/no labels.



weka.gui.GenericObjectEditor

weka.filters.unsupervised.attribute.RenameNominalValues

About

Renames the values of nominal attributes.

debug: False

doNotCheckCapabilities: False

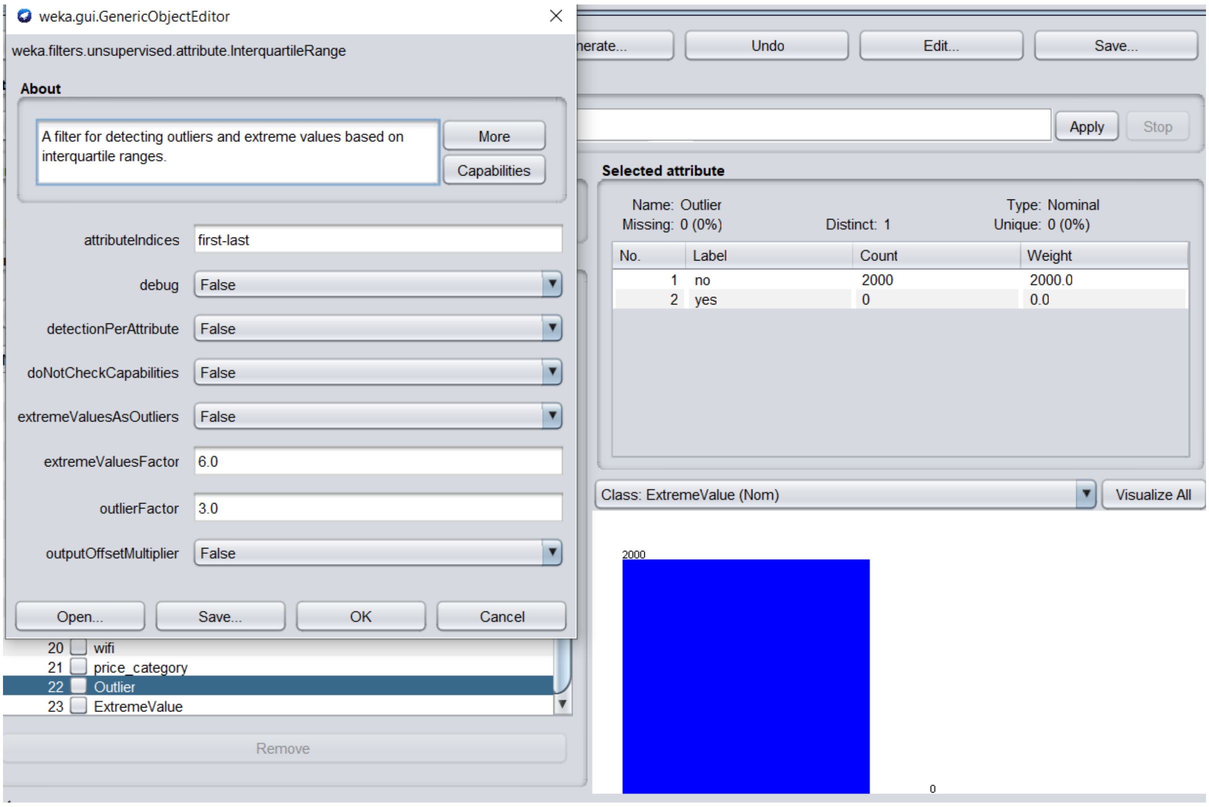
ignoreCase: False

invertSelection: False

selectedAttributes: n_cores

valueReplacements: '\'(-inf-1.875]\'':1, '\'(1.875-2.75]\'':2, '\'(2.75-3.625]\'':3, '\'(3.625-4.5

This above screenshot shows the process of replacing the labels for n_cores to represent the number of cores.



weka.gui.GenericObjectEditor

weka.filters.unsupervised.attribute.InterquartileRange

About

A filter for detecting outliers and extreme values based on interquartile ranges.

attributeIndices: first-last

debug: False

detectionPerAttribute: False

doNotCheckCapabilities: False

extremeValuesAsOutliers: False

extremeValuesFactor: 6.0

outlierFactor: 3.0

outputOffsetMultiplier: False

20 □ wifi

21 □ price_category

22 □ Outlier

23 □ ExtremeValue

Selected attribute

Name: Outlier
Missing: 0 (0%)
Distinct: 1
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	no	2000	2000.0
2	yes	0	0.0

Class: ExtremeValue (Nom)

No outliers found using the InterquartilePRange filter on the data

2. Association Rule Mining

Having ‘discretised’ all the attributes, we chose all the attributes to mine interesting patterns.

Here are the top rules for based on a metric of confidence, using a lower bound of support of 0.375 with a minimum confidence of 0.75

Rule	Confidence	Lift	Ranking

1. four_g=yes 1043 ==> three_g=yes 1043	1	1.31	All 4G phones are backwards compatible so this is not particularly interesting.
2. four_g=yes price_category=0 768 ==> three_g=yes 768	1	1.31	All 4G phones are backwards compatible so this is not particularly interesting.
3. blue=no 1010 ==> three_g=yes 782	0.77	1.02	Significantly interesting as confidence is high and lift is > 1. Also shows that non-Bluetooth phones are mostly 3G (older phones)
4. dual_sim=no 981 ==> three_g=yes 753	0.77	1.01	Significantly interesting as confidence is high and lift is > 1. It shows that dual sim is not a feature common with older 3G phones.
5. touch_screen=yes 1006 ==> three_g=yes 772	0.77	1.01	Significantly interesting as confidence is high and lift is > 1. Shows there are a large amount of touch screen phones in the data set, even with older phones.
6. wifi=yes 1014 ==> three_g=yes 774	0.76	1	Not particularly interesting that phones with Wi-Fi network capability have mobile network capability.
7. blue=no 1010 ==> price_category=0 769	0.76	1.02	Significantly interesting as confidence is high and lift is > 1. It shows that a large amount of non-Bluetooth phones is cheaper.
8. price_category=0 1500 ==> three_g=yes 1138	0.76	1	Significantly interesting as confidence is high and lift is 1. Enforces the idea that cheaper phones are 3G.
9. dual_sim=yes 1019 ==> three_g=yes 770	0.76	0.99	Moderately interesting as confidence level is above chosen to bound of 0.75 and lift is high at 1. Also, interesting that a large amount of dual sim phones are 3G.
10. touch_screen=no 994 ==> three_g=yes 751	0.76	0.99	Moderately interesting as confidence level is above chosen to bound of 0.75 and lift is high at 1. However as explored earlier we can see that there are a larger amount of touchscreen 3G phones.
11. touch_screen=yes 1006 ==> price_category=0 758	0.75	0.75	Interesting as confidence is 0.75 and lift is noticeable at 0.75 as well. It implies that touch screen is not necessarily a premium feature, that incurs higher cost.

Here are the top rules based on a metric of lift with a lower bound of support of 0.375 with a minimum lift of 1.2.

Rule	Confidence	Lift	Ranking

1. four_g=yes 1043 ==> three_g=yes 1043	1	1.31	Not as interesting as 4G phones are backwards compatible.
2. three_g=yes 1523 ==> four_g=yes 1043	0.68	1.31	Same as above.
3. four_g=yes price_category=0 768 ==> three_g=yes 768	1	1.31	Same as above.
4. three_g=yes 1523 ==> four_g=yes price_category=0 768	0.5	1.31	Same as above.
5. four_g=yes 1043 ==> three_g=yes price_category=0 768	0.74	1.29	Same as above.
6. three_g=yes price_category=0 1138 ==> four_g=yes 768	0.67	1.29	Same as above.

Explain the top k rules (according to lift or confidence) that have the "price_category" on the right-hand-side, where k >= 1.

Using a minimum support of 0.1 and a minimum confidence level of 0.8. We get the results.

1. ram='(-inf-630.2]' 215 ==> price_category=0 215   conf:(1)

2. ram='(1004.4-1378.6]' 206 ==> price_category=0 206   conf:(1)

3. ram='(1378.6-1752.8]' 200 ==> price_category=0 200   conf:(1)

4. ram='(3623.8-inf)' 208 ==> price_category=1 200   conf:(0.96)

5. ram='(2127-2501.2]' 219 ==> price_category=0 206   conf:(0.94)

6. battery_power='(-inf-650.7]' 223 ==> price_category=0 200   conf:(0.9)

7. clock_speed='(-inf-0.75]' four_g=no 255 ==> price_category=0 209   conf:(0.82)

8. sc_w='(5.4-7.2]' 262 ==> price_category=0 214   conf:(0.82)

9. sc_w='(3.6-5.4]' three_g=yes 273 ==> price_category=0 220   conf:(0.81)

10. sc_w='(3.6-5.4]' 343 ==> price_category=0 275   conf:(0.8)

Rules 1 to 5 are ram related. Phones that have RAM values of above 3623.8 are mostly in the expensive price category, whilst phones below this value are almost always (strong confidence level shows this) in the lower price category. We can infer that RAM is then a particularly important factor for which price category a phone falls in.

Rule 6 refers to phones that are in the bottom bin are mostly in the lower price category. This makes sense as mobile phones need to be mobile, so battery power is a key factor in keeping phones being able to be used wirelessly. As such phones with low battery power are going to be cheaper.

Rule 7 refers to clock speed and 4G capability in relation to price category. Essentially this rule is stating that a phone in the lowest bin of clock speed and without 4G capability is going to be in the lower price category. It is acceptable that a phone with a slow processor and slow mobile data speeds (from lack of 4G) will not be in the higher price category. (As it is older, slower components therefore less expensive).

Rules 8 through 10. These rules describe phones with screen width in the bottom bins (3rd and 4th) are likely to be in the low-price category. This rule is particularly relevant in the age of smartphones where customers are seeking larger screens rather than smaller ones.

Our Recommendation

It is evident that RAM and battery power are significant to the price of phones. Consumers are willing to pay more for phones that have higher RAM and battery power. So, a company willing to design a high-end phone should keep this in mind and design a phone with higher RAM and battery power. As customers are willing to pay for premium RAM and battery components.

3. Classification

Use the "price_category" as the target variable and train two classifiers based on different machine learning algorithms (e.g., classifier 1 based on a decision tree; classifier 2 based on SVMs).

Evaluate the classifiers based on some evaluation metrics (e.g., accuracy). You may use 10-fold cross-validation for the evaluation.

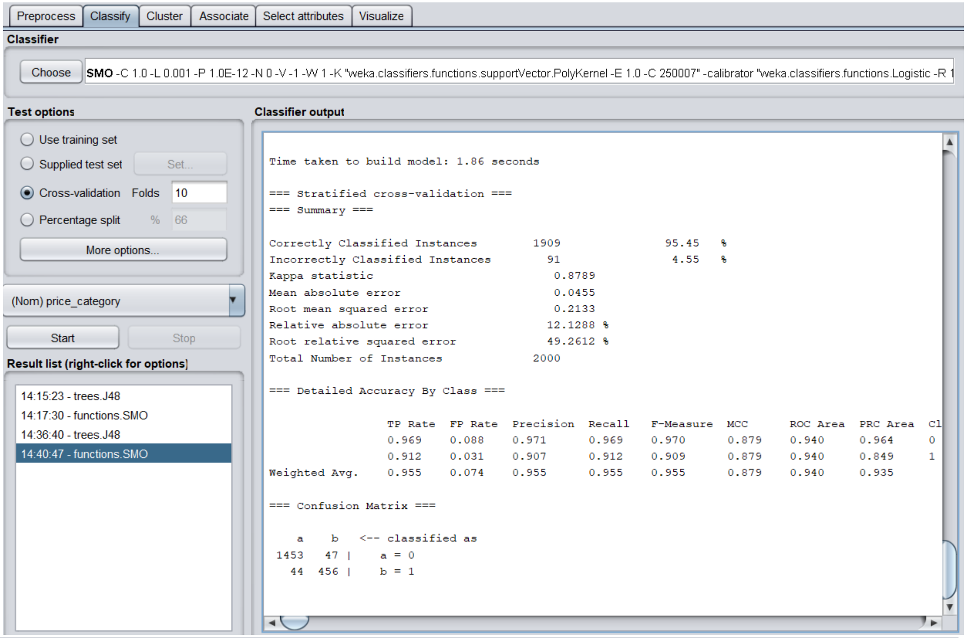
We trained 2 classifiers on the target variable of “price_category”, A J48 Decision Tree, and the SMO function algorithm (SVM). The SMO function returned a higher accuracy than the J48 Decision Tree. The decision tree returned a precision of 93.8% and 84.7% for cheap and expensive phones respectively, with a weighted average of 91.5%. However, the SMO function returned a precision of 97.1% and 90.7% for cheap and expensive phones respectively, with an impressive, weighted average of 95.5%. The SMO function shows more exactness in its results, returning a consistently higher precision for all categories.

	J48 Decision Tree	SMO function
Precision – Cheap Phones	93.8%	97.1%
Precision – Expensive Phones	84.7%	90.7%

Precision – Weighted Avg	91.5%	95.5%
Recall – Cheap Phones	95.1%	96.9%
Recall – Expensive Phones	81.0%	91.2%
Recall - Weighted Avg	91.6%	95.5%

The Recall value indicates that the SMO function also provides more completeness of the predictions than the J48 decision tree. As recall represents what % of positive instances did the classifier labels as positive. The higher percentages shown by the SMO function imply a better fit for the algorithm to this data.

Both classifiers placed RAM at the highest discerning factor for a phones price category, followed by battery power, if the RAM data was too close to discern the price category. Phones with high ram have a significantly high chance of being categorised as expensive, and phones with low ram are likely to be cheap. If the phone has a moderate amount of RAM, its price category is then decided by the size of the battery, and in very few cases, then decided on by the pixel height of the screen.



Classifier

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

Test options

○ Use training set
○ Supplied test set
● Cross-validation Folds 10
○ Percentage split % 66

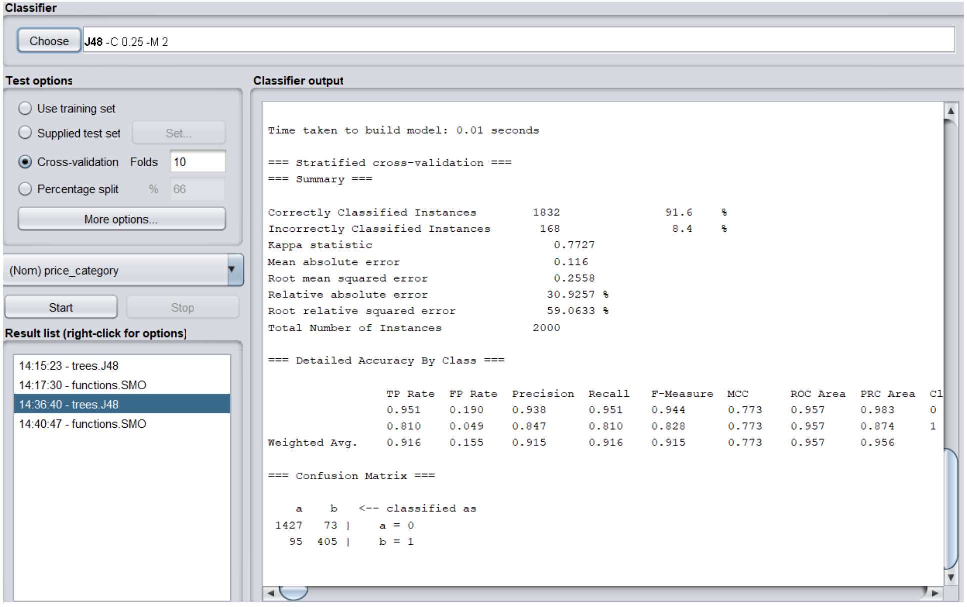
(Nom) price_category

Result list (right-click for options)

- 14:15:23 - trees.J48
- 14:17:30 - functions.SMO
- 14:36:40 - trees.J48
- 14:40:47 - functions.SMO

Classifier output

```
Time taken to build model: 1.86 seconds

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

Correctly Classified Instances        1909               95.45   %
Incorrectly Classified Instances        91                4.55   %
Kappa statistic                          0.8789
Mean absolute error                      0.0455
Root mean squared error                  0.2133
Relative absolute error                 12.1288 %
Root relative squared error             49.2612 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Cl
                 0.969    0.088    0.971      0.969    0.970      0.879    0.940     0.964     0
                 0.912    0.031    0.907      0.912    0.909      0.879    0.940     0.849     1
Weighted Avg.    0.955    0.074    0.955      0.955    0.955      0.879    0.940     0.935

=== Confusion Matrix ===

    a    b   <-- classified as
 1453   47 |    a = 0
   44  456 |    b = 1
```



Classifier

J48 -C 0.25 -M 2

Test options

○ Use training set
○ Supplied test set
● Cross-validation Folds 10
○ Percentage split % 66

(Nom) price_category

Result list (right-click for options)

- 14:15:23 - trees.J48
- 14:17:30 - functions.SMO
- 14:36:40 - trees.J48
- 14:40:47 - functions.SMO

Classifier output

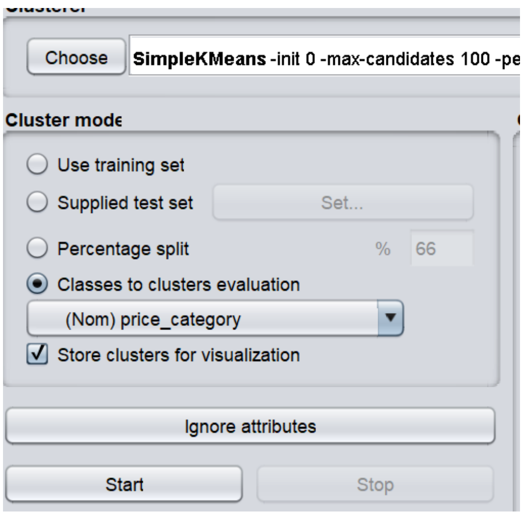
```
Time taken to build model: 0.01 seconds

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

Correctly Classified Instances        1832               91.6    %
Incorrectly Classified Instances       168                8.4    %
Kappa statistic                          0.7727
Mean absolute error                      0.116
Root mean squared error                  0.2558
Relative absolute error                 30.9257 %
Root relative squared error             59.0633 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Cl
                 0.951    0.190    0.938      0.951    0.944      0.773    0.957     0.983     0
                 0.810    0.049    0.847      0.810    0.828      0.773    0.957     0.874     1
Weighted Avg.    0.916    0.155    0.915      0.916    0.915      0.773    0.957     0.956

=== Confusion Matrix ===

    a    b   <-- classified as
 1427   73 |    a = 0
   95  405 |    b = 1
```

These were the settings used for each classifier, along with the results returned. Due to the nature of the data, the visualised tree produced was very wide and unfit to screenshot for this report, however it can be seen by opening the mobile_price_classified.arff file and pressing the visualise tree button on the tree classifier.

4. Clustering

Run a clustering algorithm of your choice and explain how the results can be interpreted with respect to the target variable.

Using Simple K-Means clustering, we clustered the data with classes to clusters evaluation based on price category (as shown in screenshot below).



Clusterer

Choose SimpleKMeans -init 0 -max-candidates 100 -pe

Cluster mode

○ Use training set

○ Supplied test set

○ Percentage split % 66

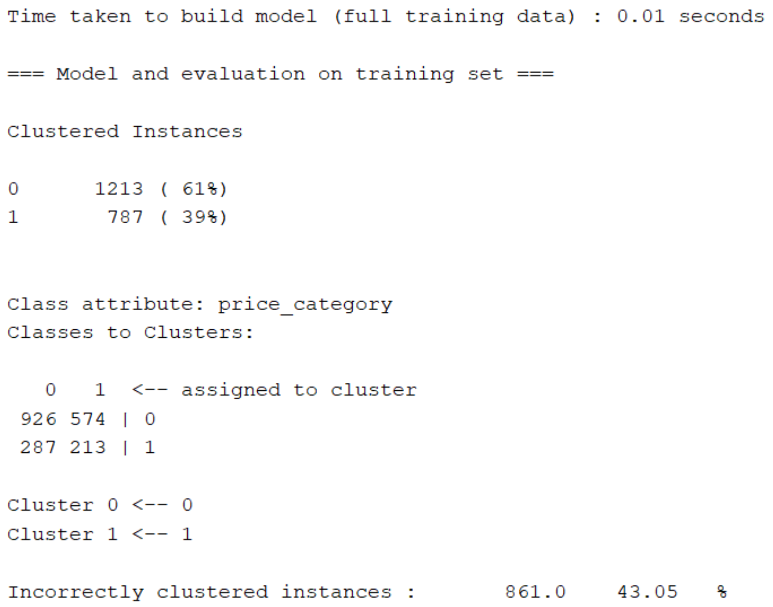
● Classes to clusters evaluation

(Nom) price_category

☑ Store clusters for visualization

We got the results:

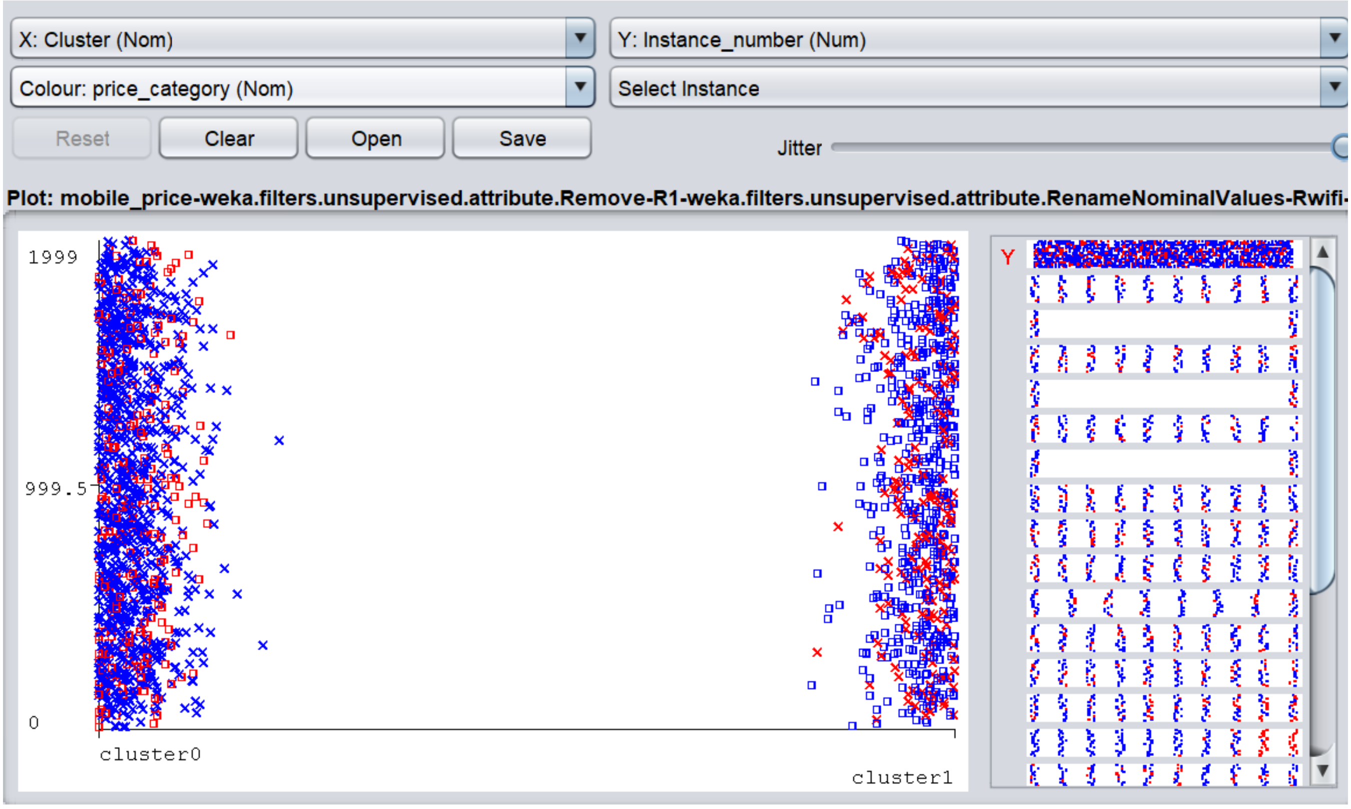
```
Final cluster centroids:
                                                Cluster#
Attribute                     Full Data                0                1
                               (2000.0)         (1213.0)          (787.0)
==========================================================================
battery_power             '(-inf-650.7]' '(1249.5-1399.2]'   '(-inf-650.7]'
blue                                 no               no              yes
clock_speed                '(-inf-0.75]'    '(-inf-0.75]'    '(-inf-0.75]'
dual_sim                            yes              yes              yes
fc                          '(-inf-1.9]'     '(-inf-1.9]'      '(1.9-3.8]'
four_g                              yes              yes              yes
int_memory                  '(-inf-8.2]'      '(26.8-33]'     '(-inf-8.2]'
m_dep                      '(-inf-0.19]'    '(-inf-0.19]'    '(-inf-0.19]'
mobile_wt                    '(-inf-92]'      '(104-116]'      '(-inf-92]'
n_cores                               4                8                6
pc                            '(-inf-2]'       '(-inf-2]'       '(18-inf)'
px_height                    '(196-392]'      '(392-588]'      '(196-392]'
px_width                  '(1848.2-inf)'  '(649.8-799.6]' '(1548.6-1698.4]'
ram                     '(2127-2501.2]'  '(2127-2501.2]' '(2875.4-3249.6]'
sc_h                         '(10.6-12]'     '(17.6-inf)'      '(10.6-12]'
sc_w                        '(-inf-1.8]'     '(-inf-1.8]'      '(1.8-3.6]'
talk_time                    '(5.6-7.4]'    '(14.6-16.4]'    '(16.4-18.2]'
three_g                             yes              yes              yes
touch_screen                        yes              yes               no
wifi                                yes              yes               no
```



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

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

Clustered Instances

0      1213 ( 61%)
1       787 ( 39%)


Class attribute: price_category
Classes to Clusters:

   0   1  <-- assigned to cluster
 926 574 | 0
 287 213 | 1

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

Incorrectly clustered instances :       861.0    43.05   %
```

The clustering returned 1213 instances in cluster 0. and 787 in cluster 1. With 861 incorrectly clustered instances. No filters were applied (excluding those applied earlier) to the data, as this gave the least incorrect clustered instances.

The significantly different attributes between the clusters are higher battery power in cluster 0, as well as higher internal memory in cluster 0. In addition, cluster 0 has taller screens but narrower screens than cluster 1. Cluster 0 also has touch screen and Wi-Fi as yes values, while 1 does not. Visualised, the data appears as follows.



X: Cluster (Nom)

Y: Instance_number (Num)

Colour: price_category (Nom)

Plot: mobile_price-weka.filters.unsupervised.attribute.Remove-R1-weka.filters.unsupervised.attribute.RenameNominalValues-Rwifi-

We can see that cluster 1 is less dense and appears to have more higher price category results. So, we can infer that price category 1 (expensive) phones are mostly in cluster 1. Which as mentioned earlier has lower battery power, internal memory, and shorter screens. However, with an error of 43.05% incorrectly placed instances, we cannot take strong inferences from these results.

5. Data Reduction.

Perform numerosity deduction and perform attribute reduction.

We performed numerosity reduction by applying the StratifiedRemoveFolds filter on the data. We deduced that 5 folds was the optimal number, leaving us with 400 instances, as opposed to the original 2000. Our attempts to utilize PCA on the attributes were unsuccessful. When not limiting the number of results, the PCA filter returned 120 attributes, which was opposite of the effect we were trying to achieve with attribute reduction, however classifier scores did improve when using these extended attributes. But when we limited the number of attributes returned by PCA to between 5-15, The classifiers returned very poor results, consistently classifying every phone as cheap. Our attempts to use the wavelet function were also unsuccessful, after package installation the option to use the wavelet filter was greyed out and attempts to apply it were unsuccessful on multiple versions of our data. Having run into inconsistent results with PCA and issues with DWT, we chose to manually remove attributes.

We chose to remove; talk time, Wi-Fi, touchscreen, mobile weight, and depth. These attributes were not found to be significant in our other analysis and after discussion we agreed that; most phones in the real world had Wi-Fi connectivity, talk time was too similar to battery power, and weight and depth were not important in determining price. For example, super light thin phones are just expensive as say a Samsung Galaxy Fold.

Train the two classifiers in task 3 on the reduced data.

After reduction, we again trained the J48 and SMO classifiers on the data. Data reduction improved the weighted average precision and recall on the J48 Decision tree classifier. The data reduction caused the J48 classifier to be more precise when classifying cheap phones, but less precise when classifying expensive phones. However, the recall percentage was the opposite, with the expensive phones' recall improving and the cheap phones' recall decreasing.

However, the data reduction decreased both precision and recall with the SMO function classifier. Precision and recall scores decreased for all categories, implying that the SMO function performs better with higher amounts of data.

	J48 Decision Tree – Pre-reduction	J48 Decision Tree – Reduced Data	SMO function - Pre-reduction	SMO function– Reduced Data
Precision – Cheap Phones	93.8%	97.6%	97.1%	94.9%
Precision – Expensive Phones	84.7%	81.6%	90.7%	81.7%
Precision – Weighted Avg	91.5%	93.6%	95.5%	91.6%
Recall – Cheap Phones	95.1%	93.0%	96.9%	93.7%
Recall – Expensive Phones	81.0%	93.0%	91.2%	85.0%
Recall - Weighted Avg	91.6%	93.0%	95.5%	91.5%

In our tests, Data reduction only improved the quality of the decision tree classifier and reduced the quality of the SMO (SVM) classifier.

Does data reduction improve the quality of the classifiers?

6. Attribute Selection

Select the top-10 most important attributes manually based on your understanding of the problem.

In no particular order, we chose, the following attributes as our most important attributes.

For the attribute of RAM, our association rule mining showed that the lower RAM value bins were in the lower price category. Leaving the bin of '3623.8-inf' (which is the highest bin). With the other lower bins, being in a different category, implying that after the threshold of 3.6 gHz phones went into the higher price category. In addition, in classification, we found RAM to be the highest discerning factor of price category. Furthermore, after data reduction and classification we found RAM to again be a key discerning factor. So, we concluded that RAM is another key indicator in 'whether the price of a mobile phone is high or not' and is therefore important.

For the attribute of battery power, our association rule mining returned. That the lowest bin of battery power was in the lower price category. In addition, our classification showed that after RAM battery power was key in discerning which price category the phone would be in. As battery power was a significant factor for 'whether the price of a mobile phone is high or not.' We view it as an important attribute.

For the attribute of pixel height, out classification found that after battery power, it was the next significant factor to determine price category (more expensive). By extension pixel width is an important attribute as well as screens sizes are determined by height and width and are not isolated from each other. So, pixel height and width are another useful factor in predicting 'whether the price of a mobile phone is high or not.'

Similar to, the aforementioned pixel resolution attributes, we determined that screen width and height were of importance. Screen width appeared in our top k association mining rules. Roughly the bottom half of bins of screen width instances were placed in the lower price category.

We also view clock speed as an important attribute. Our association rule mining returned that phones in the lowest clock speed bin and without 4G were mostly in the lower price category. In a real-world perspective, this makes sense as newer, faster processors (the component which determines clock speed) are more expensive and would place phones with higher clock speeds in the higher price category.

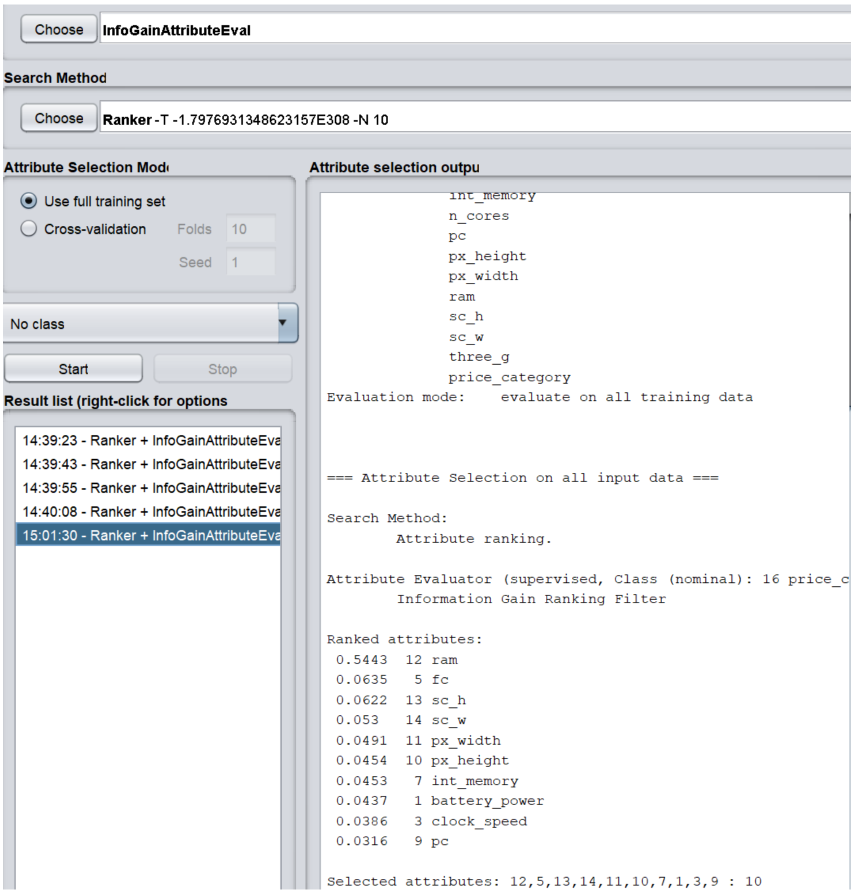
Another important attribute is Bluetooth capability. Our association rule mining should that a large amount (1010) phones that did not have Bluetooth capability were in the lower price category. This shows that Bluetooth is an essential feature of higher price category phones. And as such can be used in predicting if the price of a mobile phone is high or not.

An additional important attribute is the internal memory of the phone. This attribute was present determining the clusters. In addition, it is quite logical that a larger drive would be more expensive, as this is the nature of hardware components.

Two more important attributes are front and primary camera megapixels. These two attributes were shown to be key in determining the clusters (as well as being selected later based on information gain). A camera is a complex and expensive piece of hardware, so it is no surprise that the megapixel of both cameras significantly influences the price category.

Select the top-10 most important attributes based on Information Gain.

Using WEKA, information gain based attribute selection on the reduced data from the earlier task. we returned,


Choose
InfoGainAttributeEval
Search Method
Choose
Ranker -T -1.7976931348623157E308 -N 10
Attribute Selection Mod
● Use full training set
○ Cross-validation
Folds 10
Seed 1
No class
Start
Stop
Result list (right-click for options
14:39:23 - Ranker + InfoGainAttributeEva
14:39:43 - Ranker + InfoGainAttributeEva
14:39:55 - Ranker + InfoGainAttributeEva
14:40:08 - Ranker + InfoGainAttributeEva
15:01:30 - Ranker + InfoGainAttributeEva
Attribute selection outpu

```
              int_memory
              n_cores
              pc
              px_height
              px_width
              ram
              sc_h
              sc_w
              three_g
              price_category
Evaluation mode:    evaluate on all training data



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

Search Method:
        Attribute ranking.

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

Ranked attributes:
 0.5443  12 ram
 0.0635   5 fc
 0.0622  13 sc_h
 0.053   14 sc_w
 0.0491  11 px_width
 0.0454  10 px_height
 0.0453   7 int_memory
 0.0437   1 battery_power
 0.0386   3 clock_speed
 0.0316   9 pc

Selected attributes: 12,5,13,14,11,10,7,1,3,9 : 10
```

1. RAM
2. Front Camera megapixels
3. Screen Height
4. Screen Width
5. Pixel Resolution Height
6. Pixel Resolution Width
7. Internal Memory in Gigabytes
8. Battery Power
9. Clock Speed
10. Primary Camera mega pixels.

Which attribute selection method is better?

We concluded that selecting attributes manually is 'better'. As data scientists, we aim to use all calculations we made earlier in this report to inform, say a business decision. By critically analysing the results of pattern analysis, classification, and such we can inform our perspective on the data. As such we can report our conclusions based on different analysis techniques, to whoever wishes to 'predict whether the price of a mobile phone is high or not.' By drawing on a variety of analysis techniques rather than solely attribute selection based on information gain. We determined that, manual selection is 'better'.