

Knowledge Based Systems

ITCS 6155: PROJECT REPORT

SPRING-2017

Ashika Avula (800972702)

Bhanu Prakash Pamidi (800972464)

Sai Koumudi Kaluvakolanu (800959723)

Shanmukh Anand Gudavalli (800942628)

Vinusha Bobburu (800969202)

Table of Contents

CONTENTS	PAGE NO.
INTRODUCTION	3
DATA COLLECTION -Import.io tool -Java Code	5
PRICE DISCRETIZATION	5
MODELS USED	10
CONCLUSION	15
REFERENCES	16

Introduction:

The classification features describing the data related to fine art painting are as follows:

- Name of the Painting
- Artist Name
- Size of the Painting
- Price
- Number of Views
- Number of Favorites
- Date
- Subject
- Medium

New Features added:

We've derived five new attributes which are as follows:

- Painting Surface
 - Artist Recognition
 - Artist country
 - Shipment Mode
 - Keywords
- The aim of our project is to build our own database with all the attributes mentioned above (extracted from <http://www.saatchiart.com/paintings/fine-art>). From all the classification features, we will consider price as decision attribute and discretize it into three intervals in such a way that the resultant classifier should have highest precision.
 - In order to find which classifier has highest precision, we'll make use of few algorithms like Naive Bayes, Decision Tree, Random Forest, Logistic Regression.
 - Reasons for choosing new features in this project: We found that by adding the above five new features will improve our model and precision of our model. These five new features are more important features that describes about painting and artist which helps to build our model compared to other features. Artist recognition is important because, if the artist is featured in a collection then his/her paintings have more value or price is more for that particular painting.

Algorithms Used:

1) Naïve Bayes:

The Naive Bayesian classifier is based on Bayes' theorem with independence assumptions between predictors. The Naive Bayes algorithm is a simple probabilistic classifier that calculates a set of probabilities by counting the frequency and combinations of values in the data set. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

2) Random Forest:

A random forest is a machine learning technique used for classification, regression and feature selection. It is an ensemble technique. It means it combines the output of one weaker technique in order to get a stronger result. The weaker technique it uses is Decision tree. Random forest has a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy. It also controls over-fitting of data.

3) Decision Trees:

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is developed incrementally. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

4) Logistic Regression:

Logistic regression is also called a logit model. It is used to model dichotomous outcome variables. In the logit model the log odds of the outcome is modeled as a linear combination of the predictor variables. Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

Data Collection:

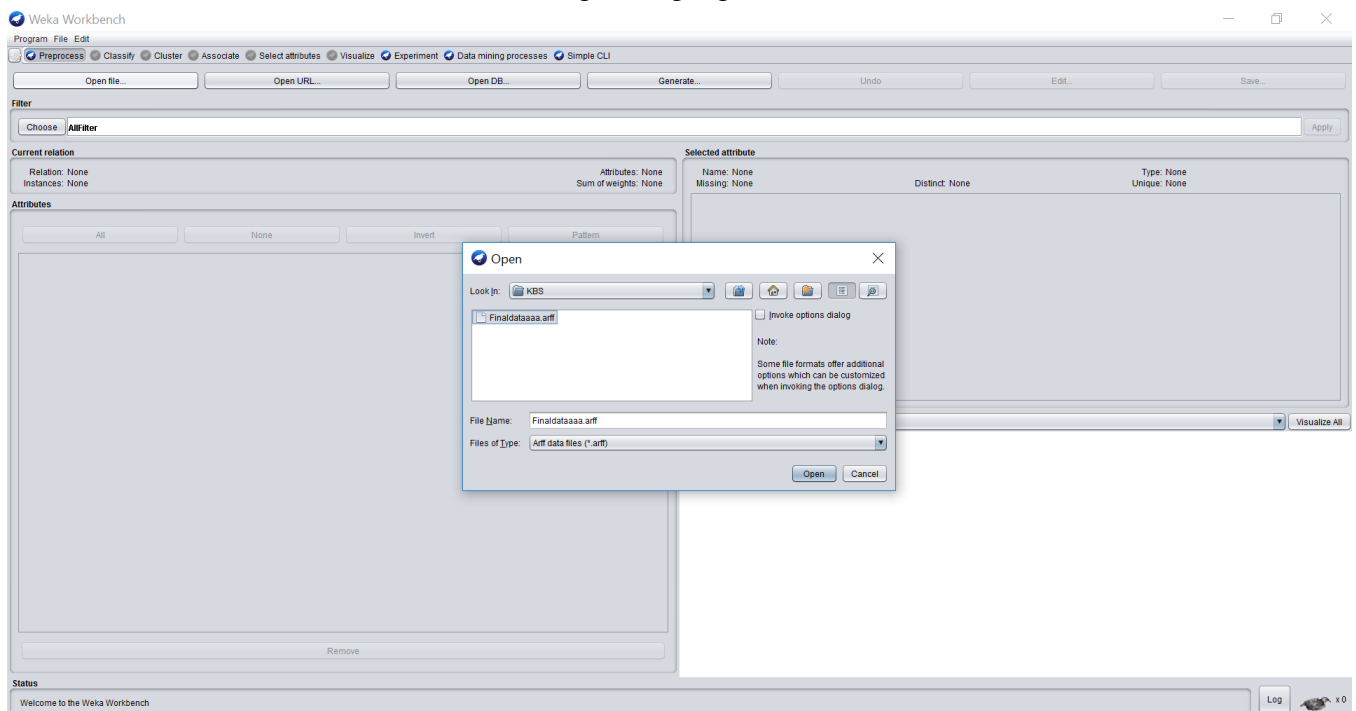
- We have used a tool called import.io to extract data related to few attributes like url, artist name, name of the painting, number of views and favorites etc., from the Saatchi art website.
- Using the url fetched using import.io, we have written a java code to fetch further attributes like medium, painting surface used, published date etc.
- The major library used in Java for web scraping is JSoup. Java code can found in the attachments.
 - Total Number of Instances: 463.
 - Attributes: 15 (Including 5 new features and url of each painting)

Price Discretization in Weka:

Screenshots of how the price attribute has been discretized into three intervals:

Step 1: Importing the .arff file into Weka.

We have converted .csv file to .arff file using R language.

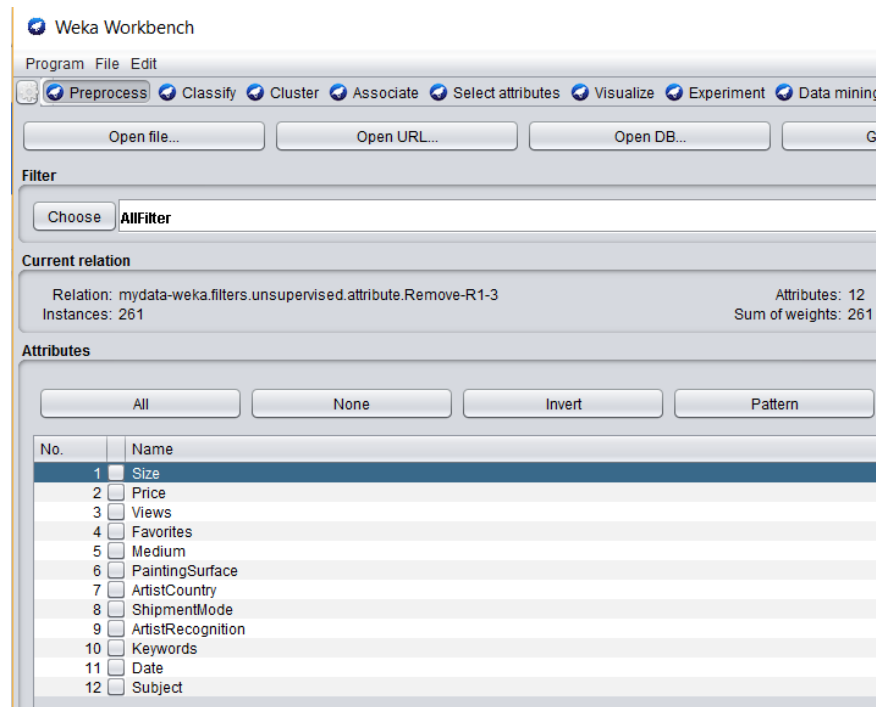


List of all 15 attributes we have extracted from the fine art paintings including new features and url of each painting:

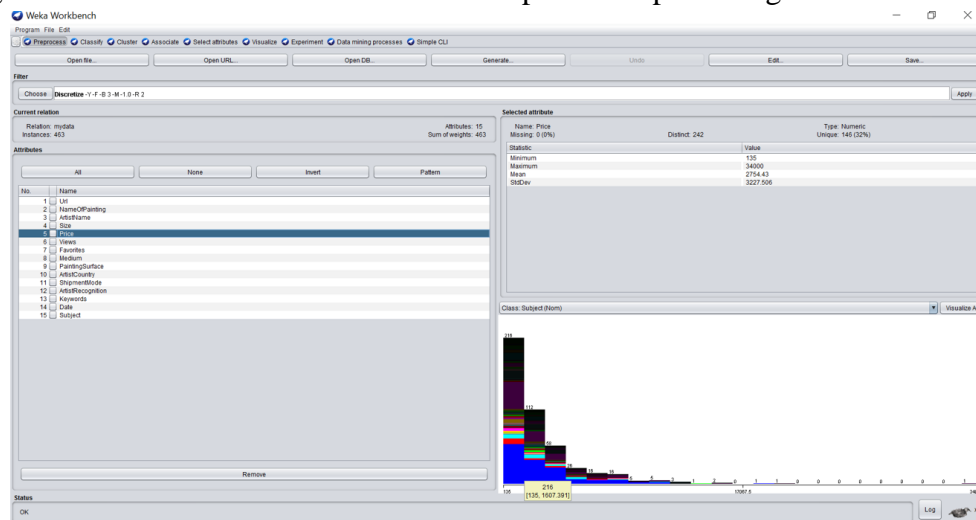
The screenshot shows a software interface with a menu bar at the top containing: Preprocess, Classify, Cluster, Associate, Select attributes, Visualize, Experiment, and Data mining pro. Below the menu bar are four buttons: Open file..., Open URL..., Open DB..., and Gene. A section labeled 'Filter' contains a 'Choose' button and a text field with 'AllFilter'. Below this is a 'Current relation' section showing 'Relation: mydata' and 'Instances: 261' on the left, and 'Attributes: 15' and 'Sum of weights: 261' on the right. The 'Attributes' section has four buttons: All, None, Invert, and Pattern. Below these buttons is a table with 15 attributes.

No.	Name
1	<input checked="" type="checkbox"/> Url
2	<input type="checkbox"/> NameOfPainting
3	<input type="checkbox"/> ArtistName
4	<input type="checkbox"/> Size
5	<input type="checkbox"/> Price
6	<input type="checkbox"/> Views
7	<input type="checkbox"/> Favorites
8	<input type="checkbox"/> Medium
9	<input type="checkbox"/> PaintingSurface
10	<input type="checkbox"/> ArtistCountry
11	<input type="checkbox"/> ShipmentMode
12	<input type="checkbox"/> ArtistRecognition
13	<input type="checkbox"/> Keywords
14	<input type="checkbox"/> Date
15	<input type="checkbox"/> Subject

Step 2:List of classifiers after removing unwanted attributes url, name of artist and painting name.

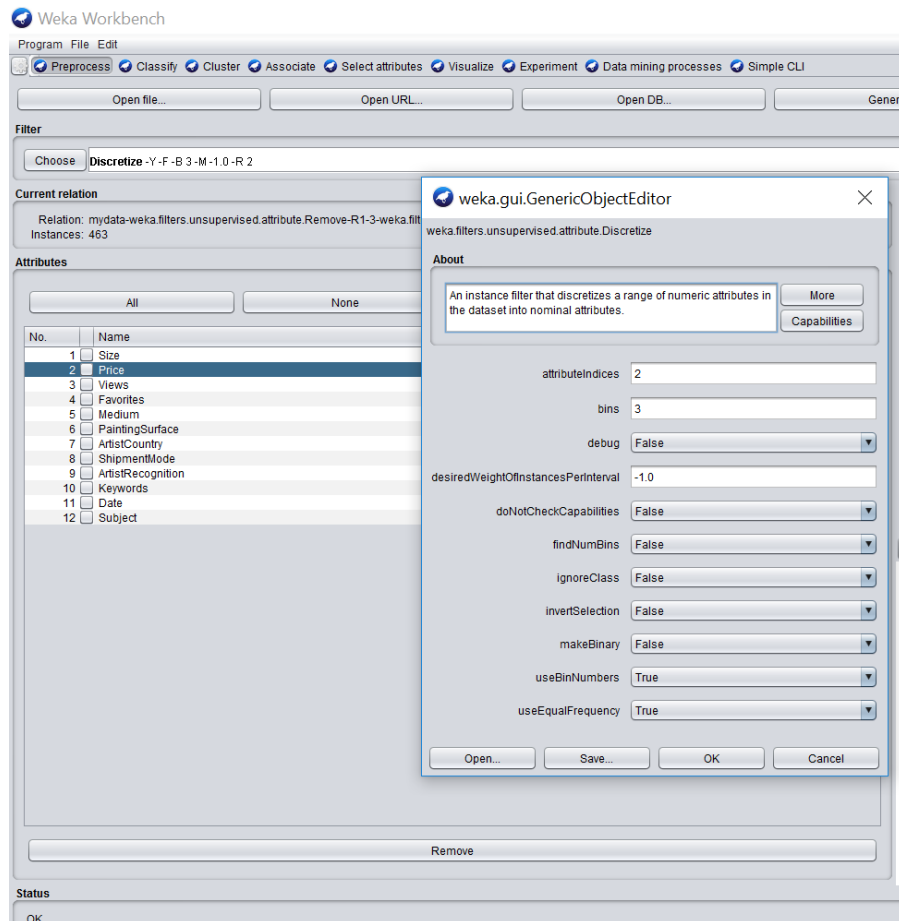


- Reason for eliminating name of painting and artist may be these two attributes are correlated or multicollinearity exists due to these attributes. If we use these two attributes model may not be significant.
- Uniqueness in price attribute is 32%.
- From the below visualization of price distribution we can say 216 instances have price from 135 - 1607.39. Price is skewed left before discretization. As we move on towards right the number of instances reduces for a particular price range.



Step 3: Discretizing price feature into 3 bins by choosing unsupervised → attributes → **Discretize** option in the preprocess tab:

- In discretization we have used Equal Frequency in order to get approximately equal distribution among the bins.

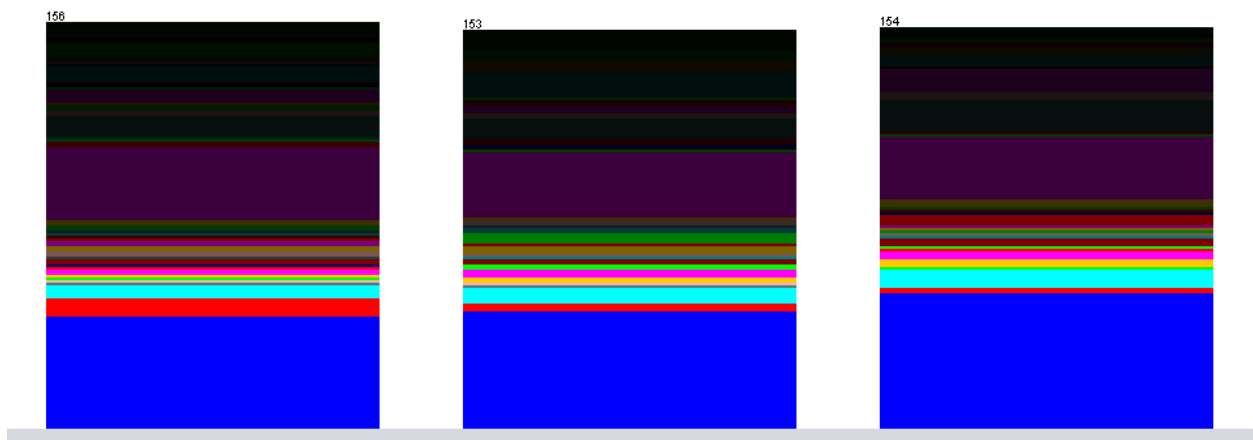


Three intervals of Price after discretization:

Selected attribute

Name: Price		Type: Nominal	
Missing: 0 (0%)		Distinct: 3	
		Unique: 0 (0%)	
No.	Label	Count	Weight
1	'B1of3'	156	156.0
2	'B2of3'	153	153.0
3	'B3of3'	154	154.0

Class: Subject (Nom) Visualize All



Step 4: After discretizing the price we used different models to check the precision.

Models Used:

- We have taken Cross-Validation Folds count as 10 for all models.

1) Naive Bayes:

- Correctly Classified Instances Precision -> 64.7948 %

The screenshot shows the Orange3 interface for the Naive Bayes classifier. The 'Test options' section on the left has 'Cross-validation' selected with 'Folds' set to 10. The 'Classifier output' pane on the right displays the following results:

```
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances    300      64.7948 %
Incorrectly Classified Instances 163      35.2052 %
Kappa statistic                  0.4756
Mean absolute error              0.2580
Root mean squared error          0.4030
Relative absolute error          55.2187 %
Root relative squared error      85.6556 %
Total Number of Instances       463

=== Detailed Accuracy By Class ===
      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
      0.821   0.182   0.686    0.821   0.753    0.436   0.896    0.777   'B1of3'
      0.491   0.197   0.531    0.491   0.488    0.266   0.700    0.519   'B2of3'
      0.469   0.149   0.691    0.469   0.480    0.524   0.865    0.768   'B3of3'
Weighted Avg. 0.468   0.176   0.640    0.468   0.441    0.470   0.821    0.689

=== Confusion Matrix ===
  a  b  c  <-- classified as
128 20  0 | a = 'B1of3'
 46 69 38 | b = 'B2of3'
 10 41 103 | c = 'B3of3'
```

2) Random Forest:

- Correctly Classified Instances Precision -> 58.9633 %

The screenshot shows the Orange3 interface for the Random Forest classifier. The 'Test options' section on the left has 'Cross-validation' selected with 'Folds' set to 10. The 'Classifier output' pane on the right displays the following results:

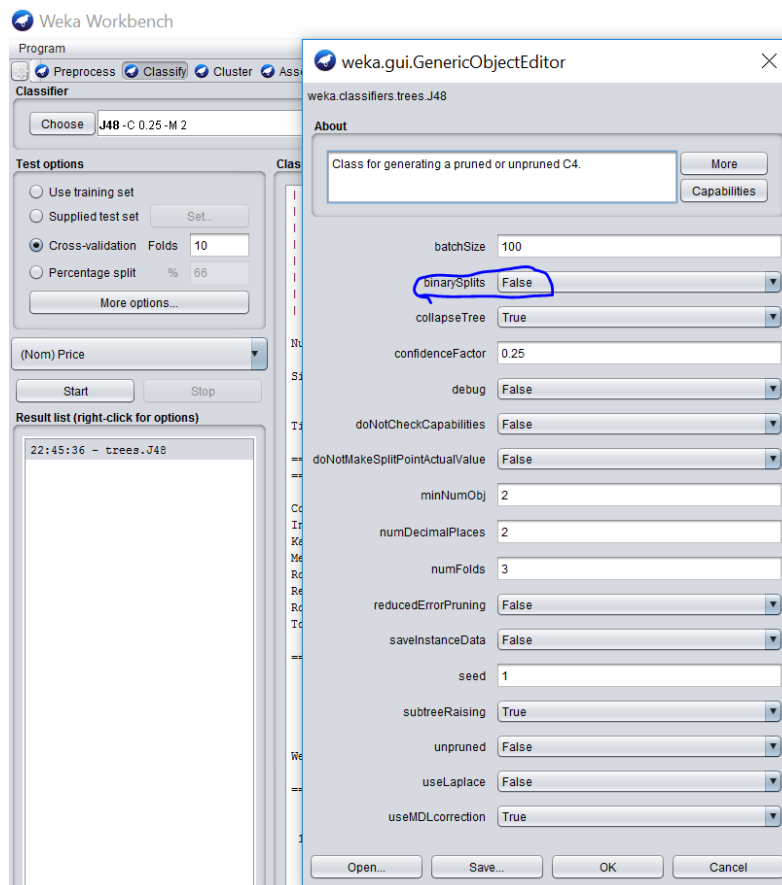
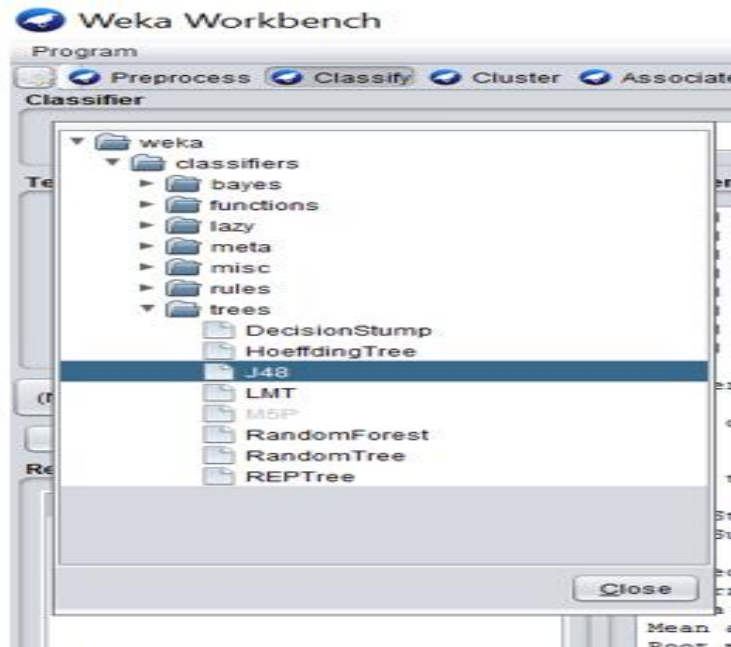
```
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances    273      58.9633 %
Incorrectly Classified Instances 190      41.0367 %
Kappa statistic                  0.3841
Mean absolute error              0.4123
Root mean squared error          0.4432
Relative absolute error          92.7576 %
Root relative squared error      94.0075 %
Total Number of Instances       463

=== Detailed Accuracy By Class ===
      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
      0.801   0.189   0.683    0.801   0.737    0.592   0.876    0.754   '(-inf-995]'
      0.386   0.219   0.465    0.386   0.421    0.175   0.626    0.452   '(995-2825]'
      0.578   0.207   0.582    0.578   0.580    0.371   0.781    0.559   '(2825-inf]'
Weighted Avg. 0.590   0.205   0.577    0.590   0.581    0.381   0.762    0.589

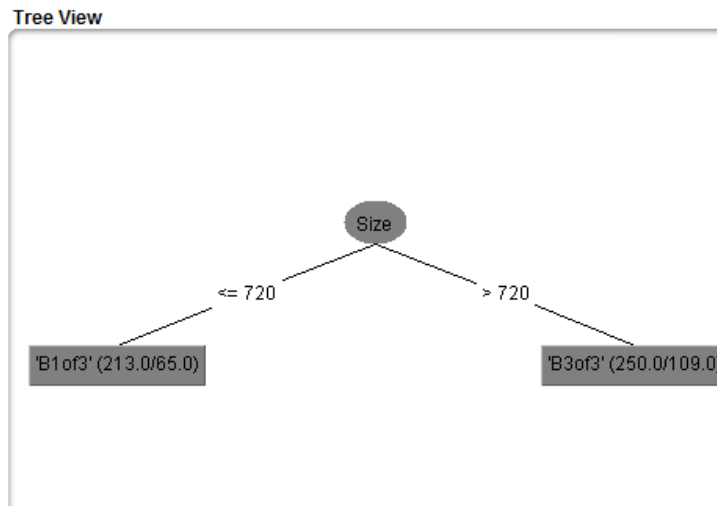
=== Confusion Matrix ===
  a  b  c  <-- classified as
125 20 11 | a = '(-inf-995]'
 41 59 53 | b = '(995-2825]'
 17 48 89 | c = '(2825-inf]'
```

3) J48:

(a) Tree when Binary Split is “False” :



Tree looks like:



- Precession value: 61.551%
- Correctly Classified Instances : 285

Weka Workbench

Program: Preprocess Classifier Cluster Associate Select attributes Visualize Experiment Data mining processes Simple CLI

Classifier: Choose J48 - C 0.25 - B - M 2

Test options: Use training set, Supplied test set, Cross-validation Folds 10, Percentage split % 65, More options...

(Norm) Price: Start, Stop

Result list (right-click for options): 22:52:10 - trees.248

Classifier output:

```

=== Classifier model (full training set) ===
J48 pruned tree
Size <= 720: 'B1of3' (213.0/65.0)
Size > 720: 'B3of3' (250.0/109.0)
Number of Leaves : 2
Size of the tree : 3
Time taken to build model: 0 seconds

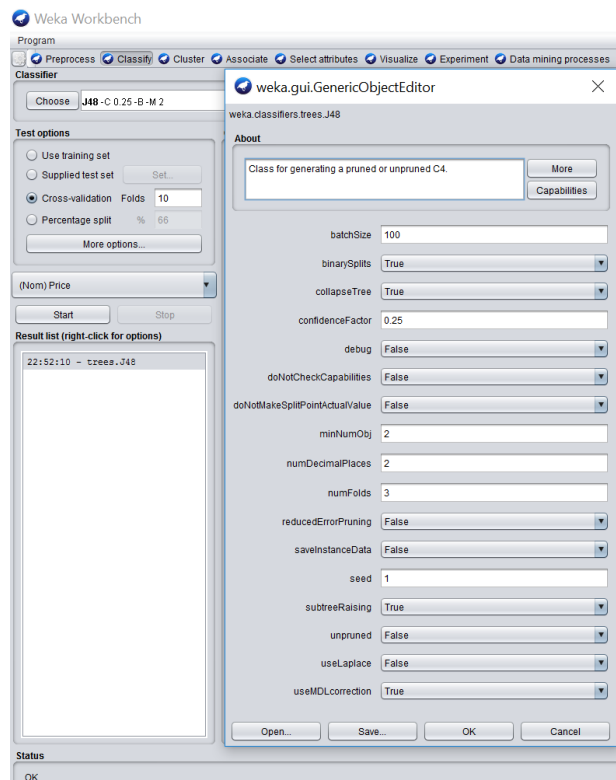
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances      285      61.551 %
Incorrectly Classified Instances    176      38.449 %
Kappa statistic                    0.4225
Mean absolute error                 0.3223
Root mean squared error             0.4063
Relative absolute error             72.524 %
Root relative squared error         86.1961 %
Total Number of Instances          463

=== Detailed Accuracy By Class ===
          TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
          0.396   0.212   0.490     0.396   0.796     0.687   0.846     0.423   'B1of3'
          0.183   0.116   0.438     0.183   0.258     0.091   0.582     0.390   'B2of3'
          0.721   0.249   0.590     0.721   0.649     0.452   0.786     0.593   'B3of3'
Weighted Avg.   0.456   0.193   0.574     0.456   0.569     0.412   0.739     0.516

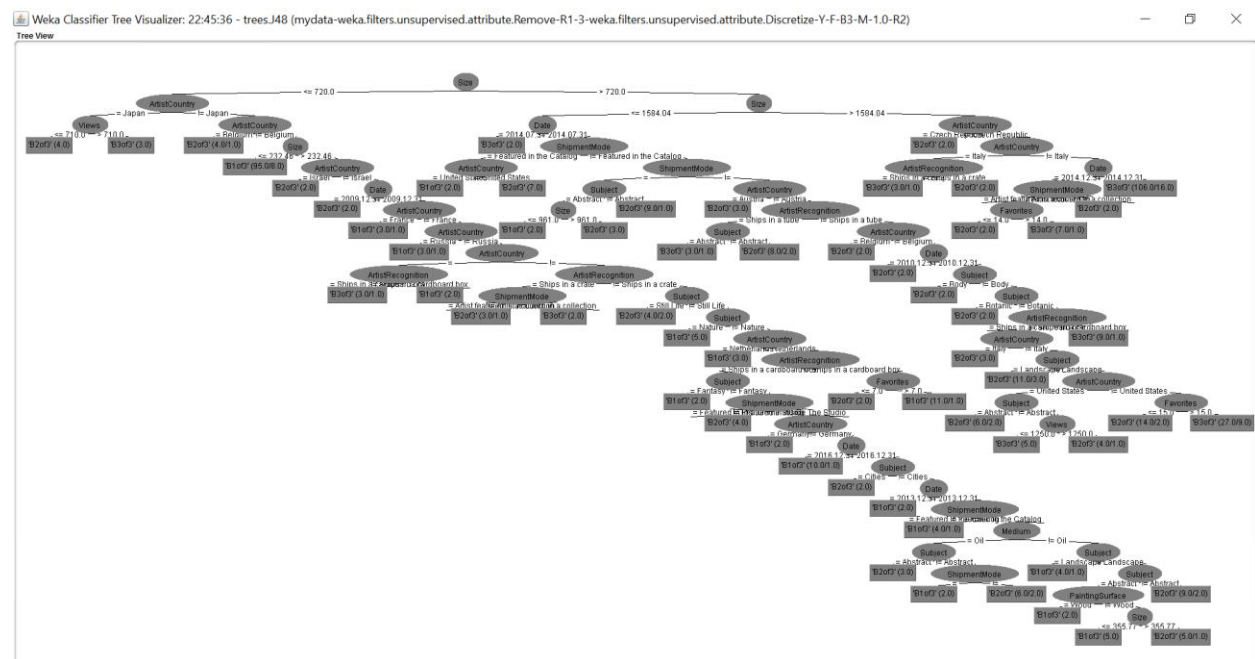
=== Confusion Matrix ===
  a  b  c  <-- classified as
146  6  4 | a = 'B1of3'
 52 28 73 | b = 'B2of3'
 13 30 111 | c = 'B3of3'
  
```

Status: OK Log

(b) When we used Binary Split = “True”



Tree Looks like:



Observations from tree:

- Root node of the tree is “size” attribute. Therefore, size is the most discriminating attribute. The second most discriminating attribute is “ArtistCountry”, which is one of the new features added. In the third stage, “ShipmentMode” is most discriminating attribute. It is also one of the new features we added.
 - Precision Value = 63.4989%
 - Correctly Classified instances 294.

The screenshot shows the Weka Workbench interface. The 'Classifier' window is active, displaying the 'Test options' and 'Classifier output' tabs. The 'Test options' tab is selected, showing 'Cross-validation' with 'Folds' set to 10. The 'Classifier output' tab shows the results of the classification process.

Classifier output:

```
| | | ArtistRecognition != Ships in a crate: 'B2of3' (2.0)
| | | ArtistCountry != Italy
| | | Date = 2014.12.31
| | | ShipmentMode = Artist featured in a collection
| | | | Favorites <= 14.0: 'B2of3' (2.0)
| | | | Favorites > 14.0: 'B2of3' (7.0/1.0)
| | | | ShipmentMode != Artist featured in a collection: 'B2of3' (2.0)
| | | | Date != 2014.12.31: 'B2of3' (106.0/16.0)
```

Number of Leaves : 60
Size of the tree : 119
Time taken to build model: 0.1 seconds

--- Stratified cross-validation ---
=== Summary ===
Correctly Classified Instances 294 63.4989 %
Incorrectly Classified Instances 169 36.5011 %
Kappa statistic 0.4522
Mean absolute error 0.2496
Root mean squared error 0.4413
Relative absolute error 60.6664 %
Root relative squared error 93.6128 %
Total Number of Instances 463

--- Detailed Accuracy By Class ---

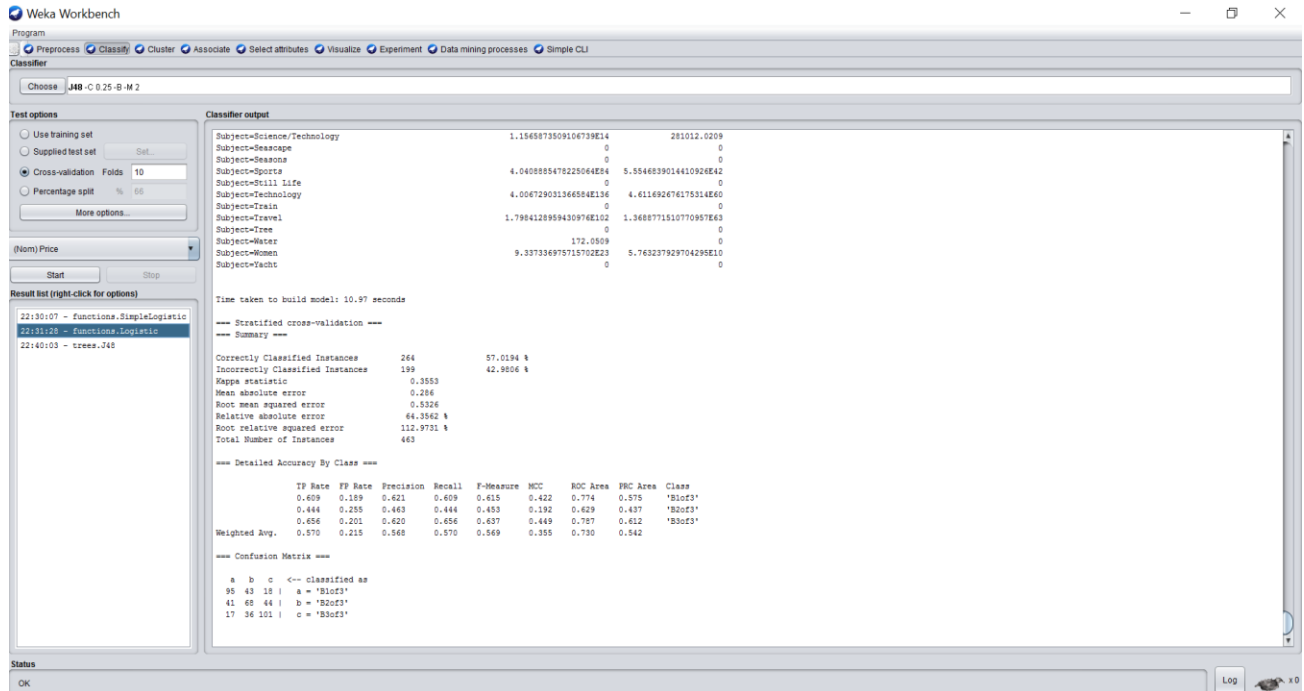
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.801	0.160	0.718	0.801	0.758	0.626	0.856	0.652	'B1of3'
	0.412	0.219	0.491	0.412	0.444	0.201	0.615	0.443	'B2of3'
	0.688	0.160	0.671	0.688	0.679	0.537	0.788	0.581	'B3of3'
Weighted Avg.	0.635	0.182	0.624	0.635	0.628	0.449	0.754	0.559	

--- Confusion Matrix ---

a \ b	c	<- classified as
125	28	a = 'B1of3'
41	63	b = 'B2of3'
8	40	c = 'B3of3'

4) Logistic Regression:

- Correctly Classified Instances - 264
- Precision -> 57.0194 %



Conclusion:

Best Model:

- From our analysis, we found that the Naive Bayes provides the best precision results. It is simple and quicker than other discriminative models like j48, Logistic Regression and Random Forest. We have implemented all the models discussed above with different number of bins and folds. From the results obtained, we found that Naive Bayes has optimal results with 3 bins and 10 folds and thus, it is the best model for our data.
- Compared to other classifiers, we found that logistic function takes more time to build the model.

Detailed accuracy from Naive Bayes:

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

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.821	0.182	0.696	0.821	0.753	0.616	0.896	0.777	'B1of3'
	0.451	0.197	0.531	0.451	0.488	0.266	0.700	0.519	'B2of3'
	0.669	0.149	0.691	0.669	0.680	0.524	0.865	0.768	'B3of3'
Weighted Avg.	0.648	0.176	0.640	0.648	0.641	0.470	0.821	0.689	

- We have also observed that, ROC area of the Naive Bayes model is more when compared to the ROC areas of other models.

- The ROC area for weighted Average for Naive Bayes is 0.821.

Confusion Matrix:

```

      a    b    c  <-- classified as
128  20    8 |   a = 'B1of3'
 46  69   38 |   b = 'B2of3'
 10  41 103 |   c = 'B3of3'

```

References:

- https://en.wikipedia.org/wiki/Naive_Bayes_classifier
- <http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
- https://en.wikipedia.org/wiki/Decision_tree_learning
- <http://ufldl.stanford.edu/tutorial/supervised/LogisticRegression/>
- <http://www.statisticssolutions.com/what-is-logistic-regression/>