Nursery Data Set

Artificial Intelligence (CPCS 331)

S2 1443H – Spring 2022

Hand in date: 18-5-2022

Team members:

Lujain Alsefri – 1805544

Sara Alahmari – 1905503

Shaikah Alroubaian - 1913619

Shaima Bashammakh – 1914892

Section: AAR

This Report Done by: Shaima Bashammakh

Task Assignment

Me and my group chose the Nursery Data Set (https://archive.ics.uci.edu/ml/datasets/Nursery), and each member in the group apply a different Machine Learning Algorithm on the chosen data set. In addition to that each member should apply her ML Algorithm in both RapidMiner and Weka softwares.

The following table shows the team members and their ML algorithm.

Student Name	The Algorithm
Lujain	Random Tree
Sara	Naïve Bayes
Shaikah	J48
Shaima	Random Forest

Table of Contents

Illustrations:

Tables:

Figures:

1. Introduction

1.1 Project Explanation

In this project, we will find the best and the most appropriate Machine Learning Algorithm that has the highest accuracy and the lowest error rate by dividing the data set into test data and training data -cross and split validations-.

1.2 Project Purpose

The purpose of the project is to learn us how to use AI approach softwares, how to implement ML algorithms on diffident softwares -RapidMiner and Weka- and how to apply these ML algorithms to a certain data set. In addition to that, the project learns us how to calculate the accuracy of the algorithms using the cross and split validations.

1.3 Outline the Approach

Firstly, we chose a certain data set from a collection of data sets, our data set is the Nursery Data Set. The chosen data set must have more than 10 000 instances. Secondly, we applied some ML algorithms on the data set using two softwares, RapidMiner and Weka. Thirdly and finally, we used cross validation and split validation to test a data set and to calculate the accuracy.

2. Technical Description

2.1 Describe the Data Set

The data set that we chose is The Nursery Data Set - click here to see it and to see its details: https://archive.ics.uci.edu/ml/datasets/Nursery -. Our data set contains 8 attributes -like parents, children and health- and 12960 instances, it also has one label -class attribute-.

The task that associated with the data set is classification. The decision that we will get from these data is to determine the class of the nursery depends on the attributes.

The attributes are:

parents: usual, pretentious, great_pret

has_nurs: proper, less_proper, improper, critical, very_crit

form: complete, completed, incomplete, foster

children: 1, 2, 3, 4 -the value 4 refers to more than 3 children-

housing: convenient, less_conv, critical

finance: convenient, inconv

social: non-prob, slightly_prob, problematic

health: recommended, priority, not_recom

2.2 Describe the ML Algorithm

The algorithm that I chose it is the Random Forest. This algorithm is one of the supervised ML algorithms that use in classification. In our data set, this algorithm helps us to find the appropriate class based on a certain number of the attributes. In addition to that, the random Forest Algorithm has an amazing feature, it can handle integer, continuous and polynomial attributes.

3. Result

3.1 Experiment Results

1- RapidMiner Results

The Algorithm	Cross Validation Accuracy (10 folds)	Split Validation Accuracy (0.7 split ratio)
Random Forest	99.00%	98.61%
Random Tree	64.69%	33.33%
Naïve Bayes	90.25%	90.41%
J48	97.85%	96.84%

Table 1: RapidMiner results

2- Weka Results

Cross Validation (10 folds):

The Algorithm	Correctly Classified Instances	Incorrectly Classified Instances
Random Forest	98.8426%	1.1574%
Random Tree	96.7052%	3.2948%
Naïve Bayes	90.2623%	9.7377%
J48	97.9398%	2.0602%

Table 2: Weka results (cross validation)

Split Validation (0.7 split ratio):

The Algorithm	Correctly Classified Instances	Incorrectly Classified Instances
Random Forest	98.534%	1.466%
Random Tree	95.1389%	4.8611%
Naïve Bayes	89.9691%	10.0309%
J48	97.3508%	2.6492%

Table 3: Weka results (split validation)

3.2 Analyze the Results

1- Analyze my algorithm (Random Forest):

After looking at the results and depending on Table1, Table2, Table3, Figure11 and Figure15 we can say that the both softwares -RapidMiner and Weka- and both validations -Cross and Split- show and give an accuracy between 89.5% and 99% so the accuracy values are extremely close on both softwares and both validations.

The best software to implement the Random Forest Algorithm is RapidMiner because it gives the highest accuracy and the lowest incorrectly classified instances.

2- Analyze the four algorithms (Random Forest, Random Tree, Naïve Bayes and J48).

After looking at the results and depending on Table1, Table2 and Table3 we can say that each algorithm has accuracy values close to each other on both softwares -RapidMiner and Weka- and both validations -Cross and Split- except the Random Tree Algorithm.

In the Random Tree Algorithm, the accuracy value on the RapidMiner software is much less than the accuracy value on the Weka software, and also the split validation on RapidMiner software is much less than the cross validation.

The accuracy of the Random Forest Algorithm is between 89.5% and 99%

The accuracy of the Random Tree Algorithm is between 33.3% and 96.8%

The accuracy of the Naïve Bayes Algorithm is between 89.9% and 90.5%

The accuracy of the J48 Algorithm is between 96.9% and 98%

4. Conclusion

After comparing and analysis the algorithms of the team members, we concluded that the Random Forest Algorithm is the most accurate algorithm of nursery data set, it has the highest accuracy and its accuracy reached 99.00%

5. References

1- UCI machine learning repository: Nursery data set. (n.d.). https://archive.ics.uci.edu/ml/datasets/Nursery

2- Random forest | Introduction to random forest algorithm. (2021, June 24). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

6. Appendices

6.1 RapidMiner Software

The Cross Validation:

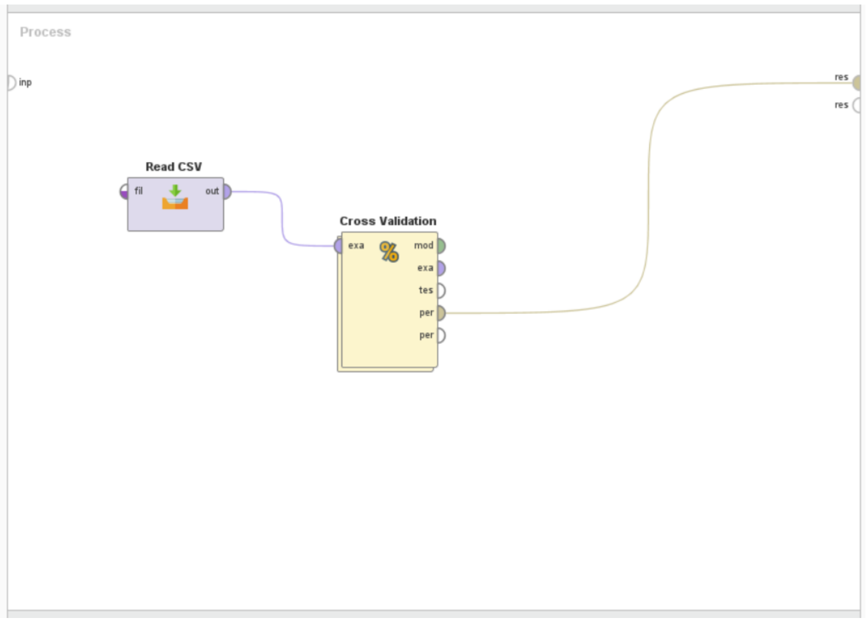


Figure 1: A screenshot of RapidMiner (cross validation)

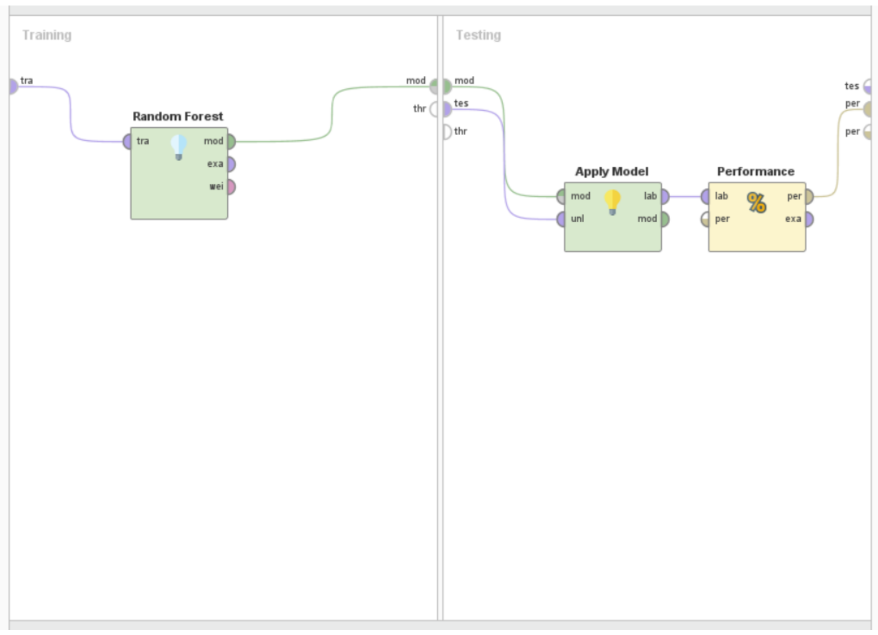
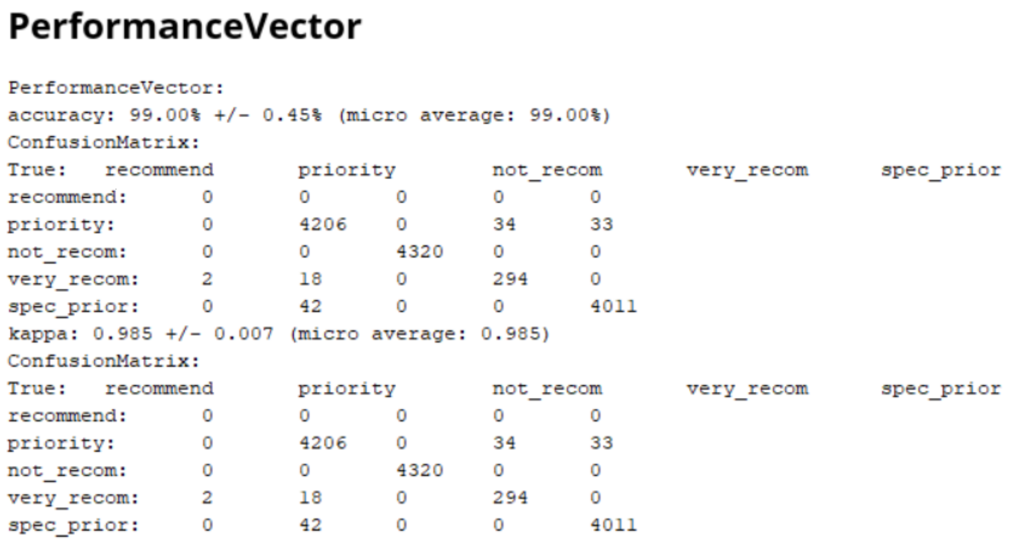


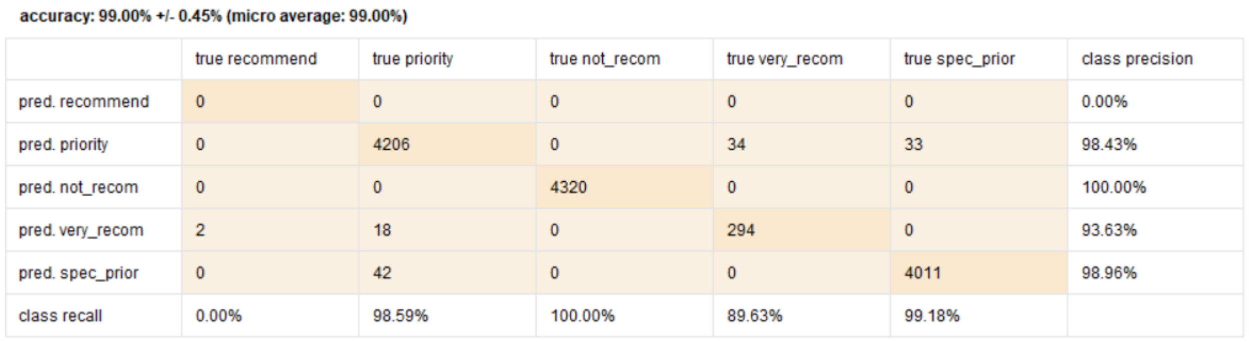
Figure 2: A screenshot of cross validation tool with 10 folds



PerformanceVector

```
PerformanceVector:
accuracy: 99.00% +/- 0.45% (micro average: 99.00%)
ConfusionMatrix:
True:   recommend       priority        not_recom       very_recom      spec_prior
recommend:      0       0       0       0       0
priority:       0       4206    0       34      33
not_recom:      0       0       4320    0       0
very_recom:     2       18      0       294     0
spec_prior:     0       42      0       0       4011
kappa: 0.985 +/- 0.007 (micro average: 0.985)
ConfusionMatrix:
True:   recommend       priority        not_recom       very_recom      spec_prior
recommend:      0       0       0       0       0
priority:       0       4206    0       34      33
not_recom:      0       0       4320    0       0
very_recom:     2       18      0       294     0
spec_prior:     0       42      0       0       4011
```

Figure 3: The performance description of RapidMiner (cross validation)



accuracy: 99.00% +/- 0.45% (micro average: 99.00%)

	true recommend	true priority	true not_recom	true very_recom	true spec_prior	class precision
pred. recommend	0	0	0	0	0	0.00%
pred. priority	0	4206	0	34	33	98.43%
pred. not_recom	0	0	4320	0	0	100.00%
pred. very_recom	2	18	0	294	0	93.63%
pred. spec_prior	0	42	0	0	4011	98.96%
class recall	0.00%	98.59%	100.00%	89.63%	99.18%	

Figure 4: The accuracy of RapidMiner (cross validation)

The Split Validation:

1- The Split Validation with 0.7 Split Ratio:

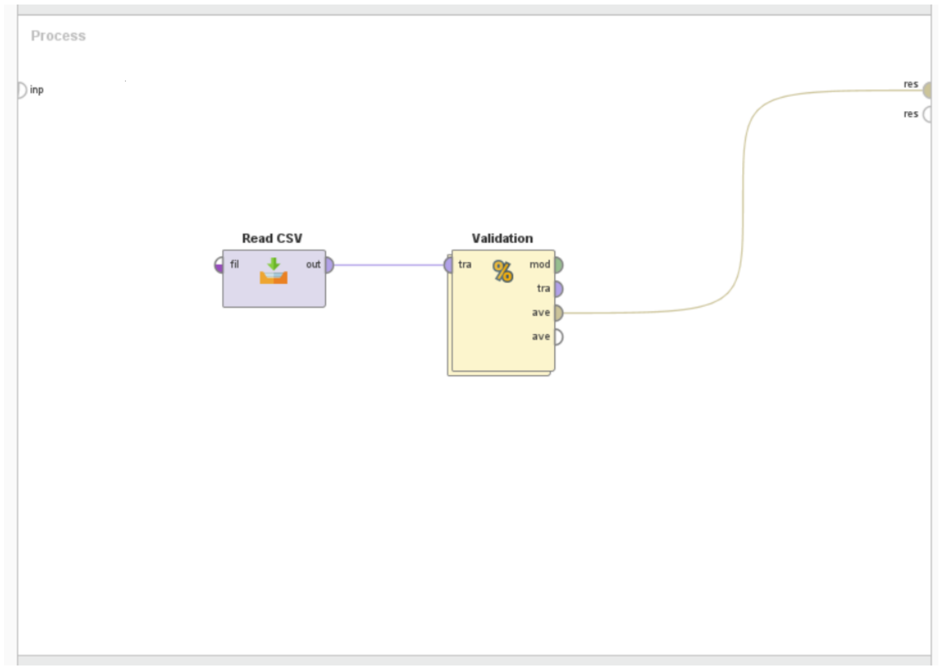


Figure 5: A screenshot of RapidMiner (split validation with 0.7 split ratio)

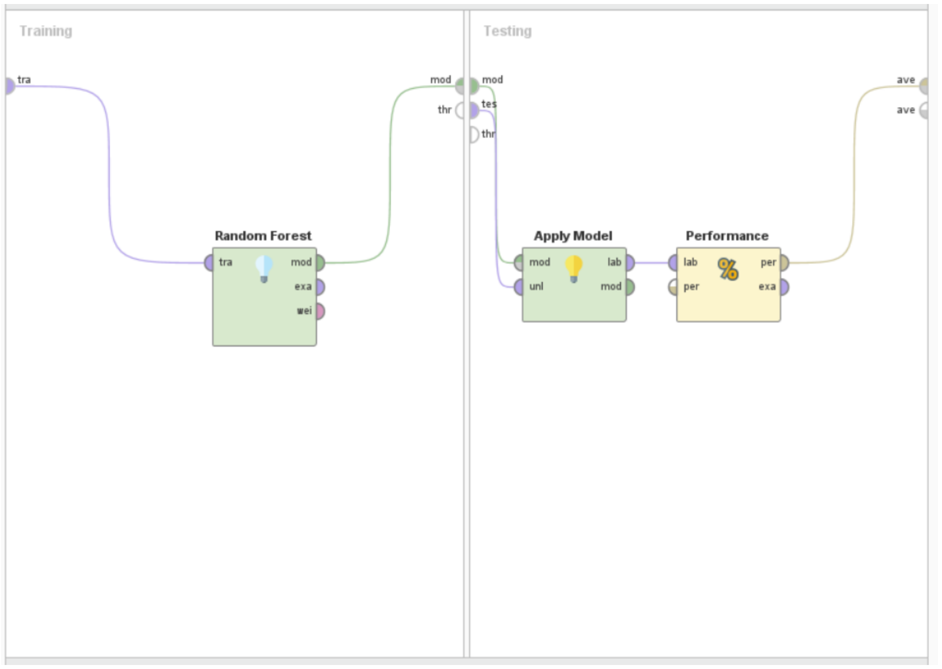
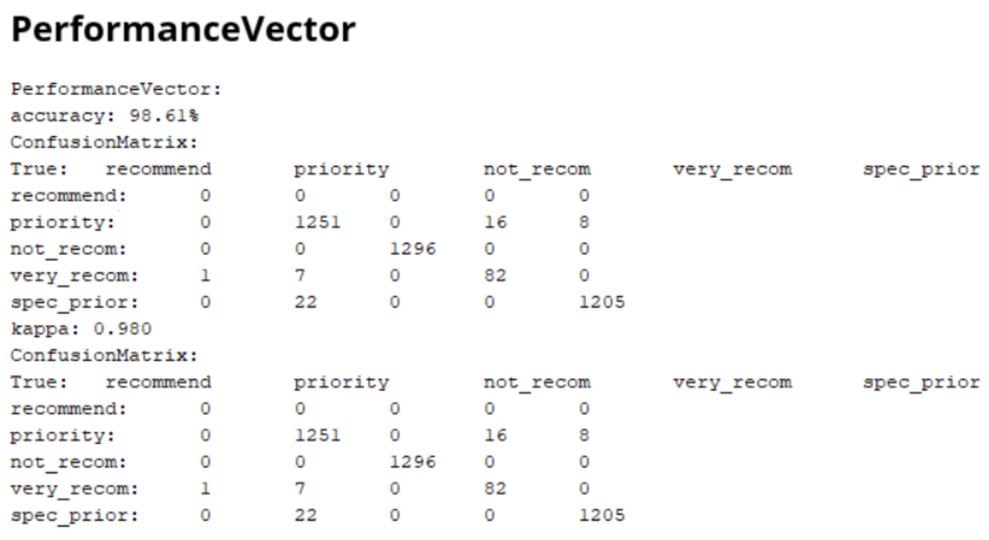


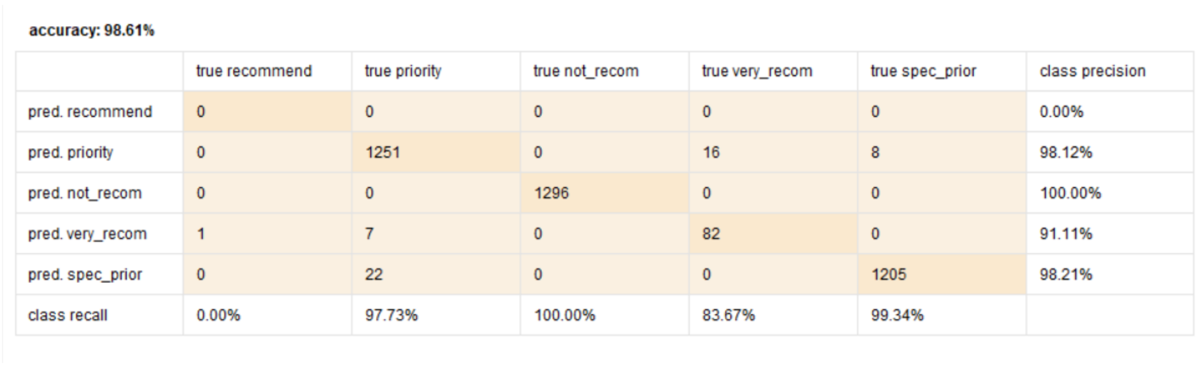
Figure 6: A screenshot of split validation tool with 0.7 split ratio



PerformanceVector

```
PerformanceVector:
accuracy: 98.61%
ConfusionMatrix:
True:   recommend       priority        not_recom       very_recom      spec_prior
recommend:      0       0       0       0       0
priority:       0       1251    0       16      8
not_recom:      0       0       1296    0       0
very_recom:     1       7       0       82      0
spec_prior:     0       22      0       0       1205
kappa: 0.980
ConfusionMatrix:
True:   recommend       priority        not_recom       very_recom      spec_prior
recommend:      0       0       0       0       0
priority:       0       1251    0       16      8
not_recom:      0       0       1296    0       0
very_recom:     1       7       0       82      0
spec_prior:     0       22      0       0       1205
```

Figure 7: The performance description of RapidMiner (split validation with 0.7 split ratio)



accuracy: 98.61%

	true recommend	true priority	true not_recom	true very_recom	true spec_prior	class precision
pred. recommend	0	0	0	0	0	0.00%
pred. priority	0	1251	0	16	8	98.12%
pred. not_recom	0	0	1296	0	0	100.00%
pred. very_recom	1	7	0	82	0	91.11%
pred. spec_prior	0	22	0	0	1205	98.21%
class recall	0.00%	97.73%	100.00%	83.67%	99.34%	

Figure 8: The accuracy of RapidMiner (split validation with 0.7 split ratio)

2- The Split Validation with 0.8 Split Ratio:

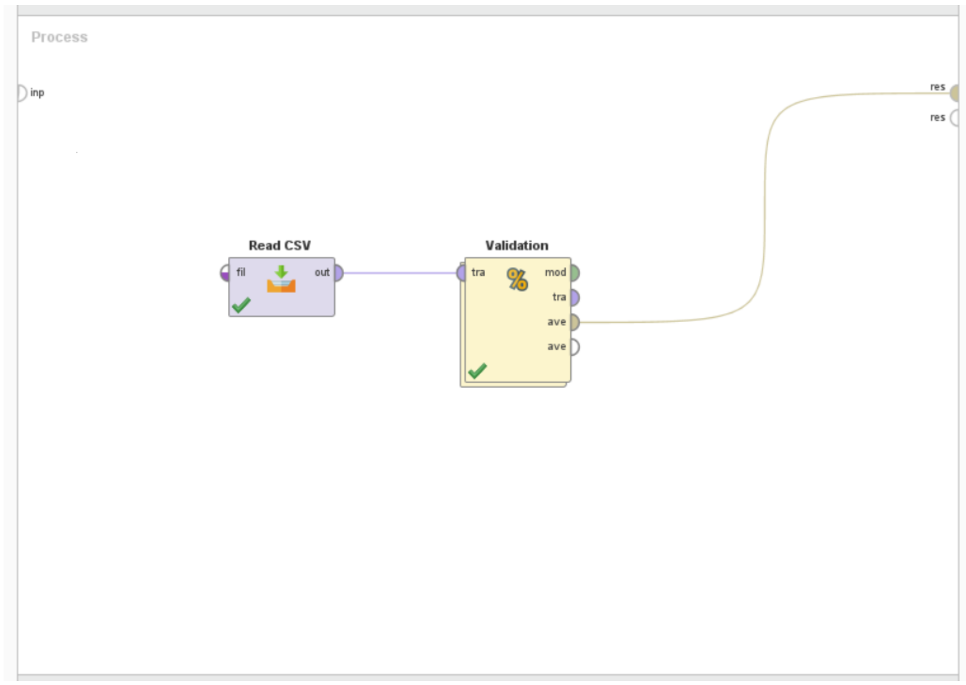


Figure 9: A screenshot of RapidMiner (split validation with 0.8 split ratio)

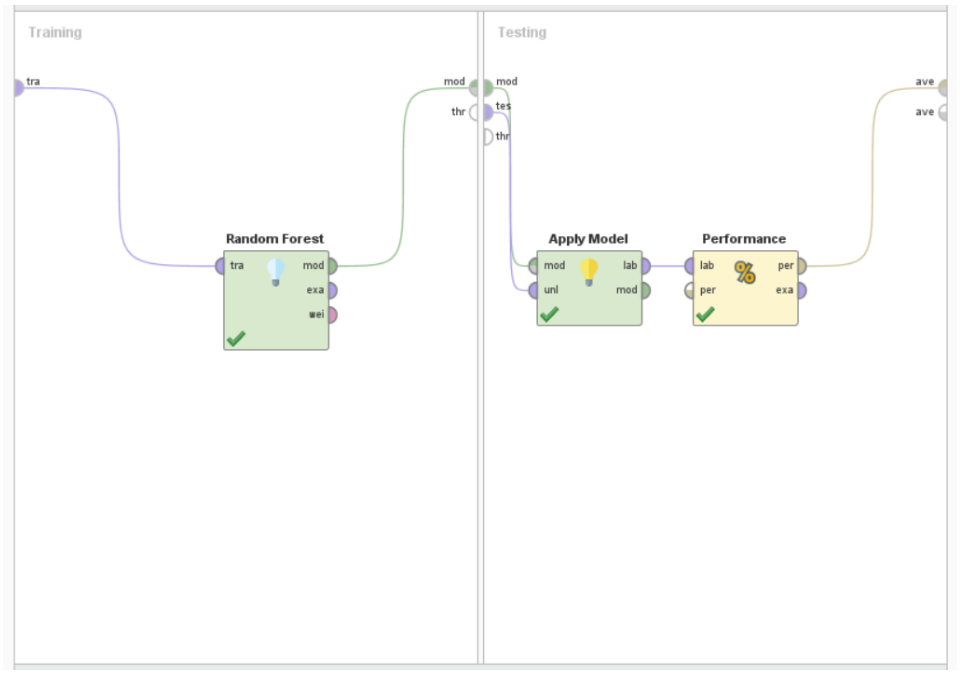
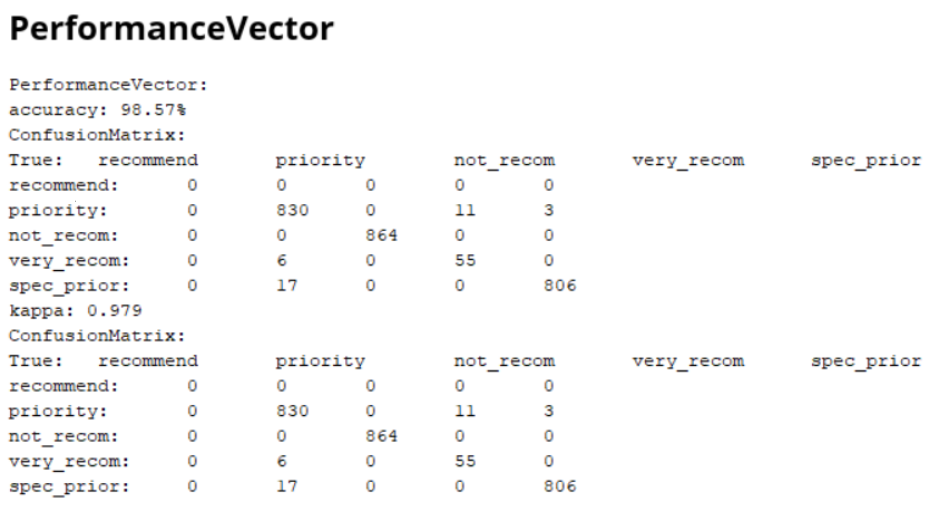


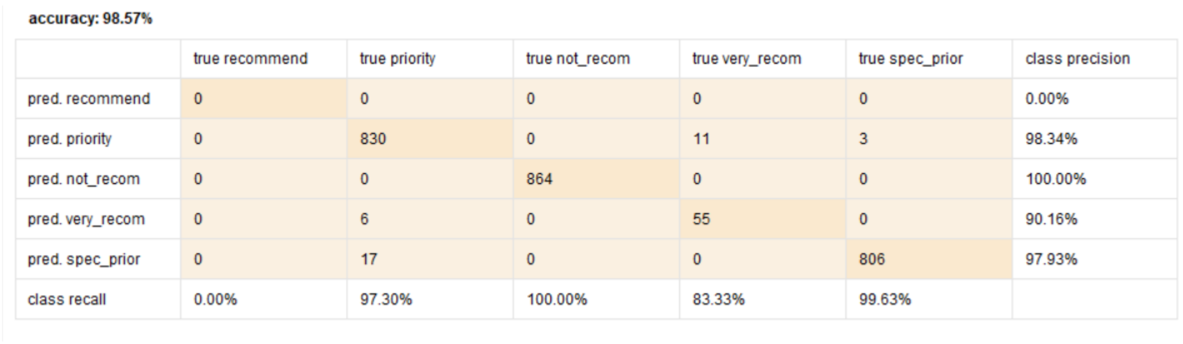
Figure 10: A screenshot of split validation tool with 0.8 split ratio



PerformanceVector

```
PerformanceVector:
accuracy: 98.57%
ConfusionMatrix:
True:   recommend       priority        not_recom       very_recom      spec_prior
recommend:      0       0       0       0       0
priority:       0       830     0       11      3
not_recom:      0       0       864     0       0
very_recom:     0       6       0       55      0
spec_prior:     0       17      0       0       806
kappa: 0.979
ConfusionMatrix:
True:   recommend       priority        not_recom       very_recom      spec_prior
recommend:      0       0       0       0       0
priority:       0       830     0       11      3
not_recom:      0       0       864     0       0
very_recom:     0       6       0       55      0
spec_prior:     0       17      0       0       806
```

Figure 11: The performance description of RapidMiner (split validation with 0.8 split ratio)



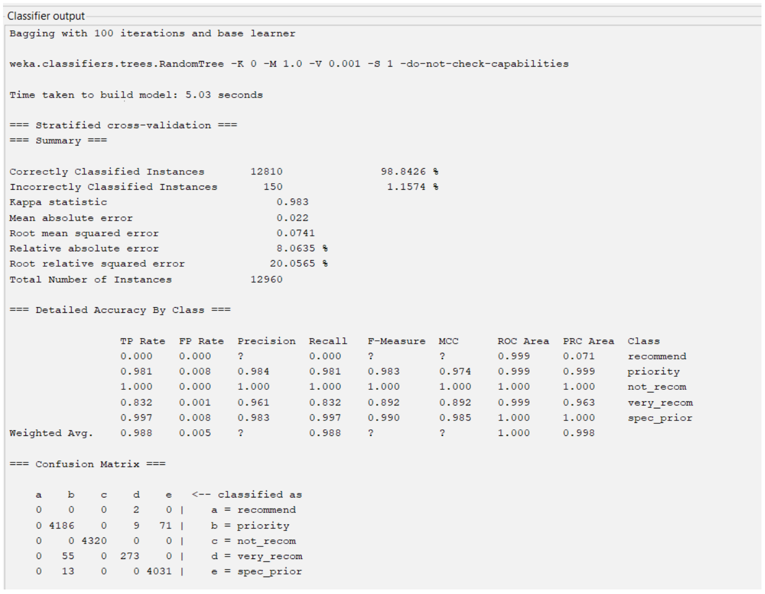
accuracy: 98.57%

	true recommend	true priority	true not_recom	true very_recom	true spec_prior	class precision
pred. recommend	0	0	0	0	0	0.00%
pred. priority	0	830	0	11	3	98.34%
pred. not_recom	0	0	864	0	0	100.00%
pred. very_recom	0	6	0	55	0	90.16%
pred. spec_prior	0	17	0	0	806	97.93%
class recall	0.00%	97.30%	100.00%	83.33%	99.63%	

Figure 12: The accuracy of RapidMiner (split validation with 0.8 split ratio)

6.2 Weka Software

The Cross Validation:



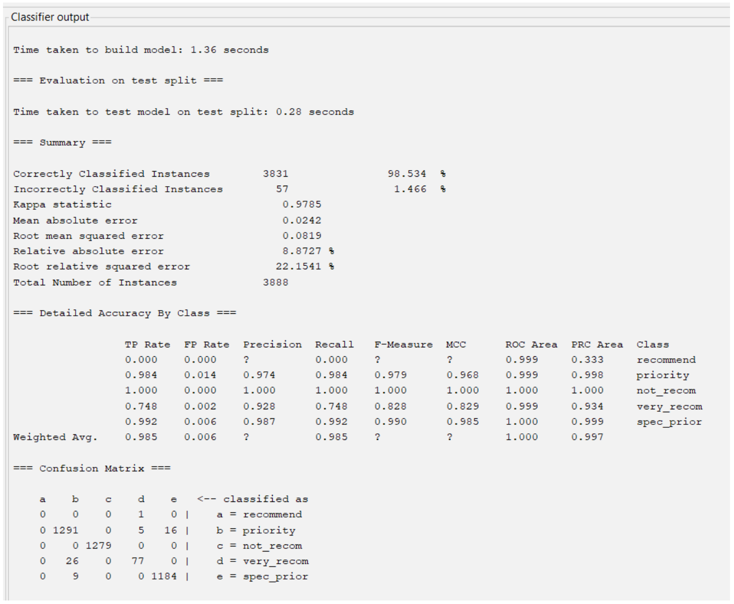
Classifier output

```
Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 5.03 seconds

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

Correctly Classified Instances       12810               98.8426 %
Incorrectly Classified Instances       150                1.1574 %
Kappa statistic                          0.983
Mean absolute error                      0.022
Root mean squared error                  0.0741
Relative absolute error                  8.0635 %
Root relative squared error             20.0565 %
Total Number of Instances            12960

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.000    0.000    ?          0.000    ?          ?        0.999     0.071     recommend
                 0.981    0.008    0.984      0.981    0.983      0.974    0.999     0.999     priority
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     not_recom
                 0.832    0.001    0.961      0.832    0.892      0.892    0.999     0.963     very_recom
                 0.997    0.008    0.983      0.997    0.990      0.985    1.000     1.000     spec_prior
Weighted Avg.    0.988    0.005    ?          0.988    ?          ?        1.000     0.998

=== Confusion Matrix ===

    a    b    c    d    e   <-- classified as
    0    0    0    2    0 |    a = recommend
    0 4186    0    9   71 |    b = priority
    0    0 4320    0    0 |    c = not_recom
    0   55    0  273    0 |    d = very_recom
    0   13    0    0 4031 |    e = spec_prior
```

Figure 13: A screenshot of Weka (cross validation)

The Split Validation:

1- The Split Validation with 70% Percentage Split:

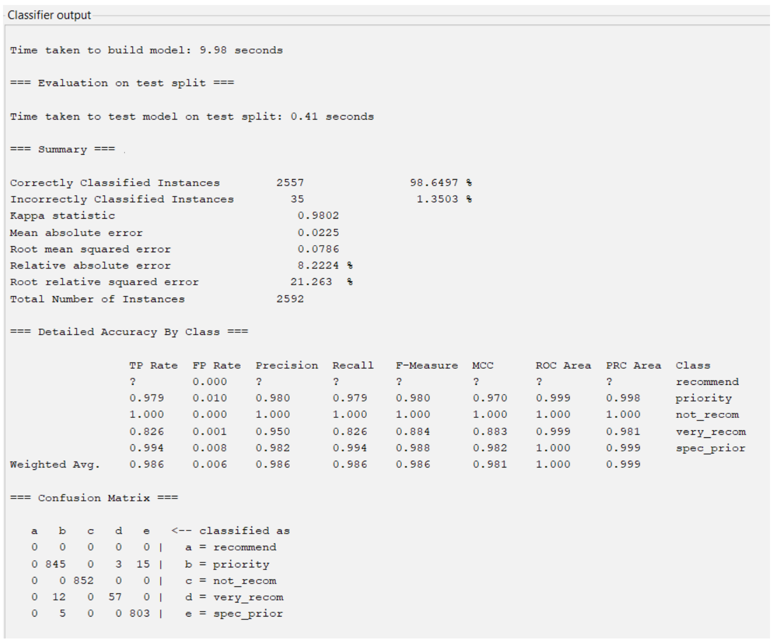


Classifier output

```
Time taken to build model: 1.36 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.28 seconds

=== Summary ===

Correctly Classified Instances        3831               98.534  %
Incorrectly Classified Instances        57                1.466  %
Kappa statistic                          0.9785
Mean absolute error                      0.0242
Root mean squared error                  0.0819
Relative absolute error                  8.8727 %
Root relative squared error             22.1541 %
Total Number of Instances             3888

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.000    0.000    ?          0.000    ?          ?        0.999     0.333     recommend
                 0.984    0.014    0.974      0.984    0.979      0.968    0.999     0.998     priority
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     not_recom
                 0.748    0.002    0.928      0.748    0.828      0.829    0.999     0.934     very_recom
                 0.992    0.006    0.987      0.992    0.990      0.985    1.000     0.999     spec_prior
Weighted Avg.    0.985    0.006    ?          0.985    ?          ?        1.000     0.997

=== Confusion Matrix ===

    a    b    c    d    e   <-- classified as
    0    0    0    1    0 |    a = recommend
    0 1291    0    5   16 |    b = priority
    0    0 1279    0    0 |    c = not_recom
    0   26    0   77    0 |    d = very_recom
    0    9    0    0 1184 |    e = spec_prior
```

Figure 14: A screenshot of Weka (split validation with 70 percentage split)

2- The Split Validation with 80% Percentage Split:



```
Time taken to build model: 9.98 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.41 seconds

=== Summary ===

Correctly Classified Instances        2557               98.6497 %
Incorrectly Classified Instances        35                1.3503 %
Kappa statistic                          0.9802
Mean absolute error                      0.0225
Root mean squared error                  0.0786
Relative absolute error                  8.2224 %
Root relative squared error             21.263  %
Total Number of Instances             2592

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 ?        0.000    ?          ?        ?          ?        ?         ?         recommend
                 0.979    0.010    0.980      0.979    0.980      0.970    0.999     0.998     priority
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     not_recom
                 0.826    0.001    0.950      0.826    0.884      0.883    0.999     0.981     very_recom
                 0.994    0.008    0.982      0.994    0.988      0.982    1.000     0.999     spec_prior
Weighted Avg.    0.986    0.006    0.986      0.986    0.986      0.981    1.000     0.999

=== Confusion Matrix ===

   a   b   c   d   e   <-- classified as
   0   0   0   0   0 |   a = recommend
   0 845   0   3  15 |   b = priority
   0   0 852   0   0 |   c = not_recom
   0  12   0  57   0 |   d = very_recom
   0   5   0   0 803 |   e = spec_prior
```

Figure 15: A screenshot of Weka (split validation with 80 percentage split)