Model Performance and Recommendation Systems

Sridhar Seshadri

Overview

Performance Evaluation of Classifiers

Recommendation Systems

Model Performance: Bias vs Variance

Use all the available information in dataset to build prediction model.

However, using too much information from the current data -- may not be a good idea as this may lead to poor prediction outside the training data.

Model Performance: Bias vs Variance

Thus, to ensure that we have not fit our model to the training data, it's a good idea to test our model performance (prediction accuracy) on a validation dataset.

Therefore, we do not use the validation dataset in fitting the model, but we do use it to choose the final model.

Model Performance: Test

To evaluate the performance of our final model (fitted on training data and selected on the basis of performance on the validation data), we may also keep a holdout sample (test data).

Model Performance: Test

Data is divided into three parts: train, validation, and test.

We must test our model accuracy, error rates, etc. on the validation set. (Rattle warns!)

Model Performance: Cross-Validate

Reducing the problem of overfitting (high variance) through cross-validation

Partition the dataset into, say, ten subsets.

Any nine of these ten form the training set and the last one is the hold-out set to measure model performance.

Model Performance: Cross-Validate

This process can be repeated ten times.

The model performance across these ten repetitions can be averaged to get expected model performance.

Metrics to Evaluate Model Performance

Error rate

True and False Positives and Negatives

Precision, Recall, Sensitivity, Specificity

Confusion matrix

Risk charts

Lift chart

Scoring

Classification Tree – Spam filter

Needs DAAG package

Objective – To classify emails as spam vs non-spam.

crl.tot total length of words in capitals

dollar number of occurrences of the \$ symbol

bang number of occurrences of the ! symbol

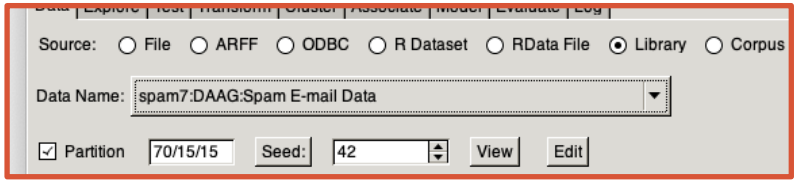
money number of occurrences of the word 'money'

n000 number of occurrences of the string '000'

make number of occurrences of the word 'make'

yesno outcome variable, a factor with levels n not spam, y spam

Source- DAAG package library R
Original source - http://archive.ics.uci.edu/ml/machine-learning-databases/spambase/



crl.tot	dollar	bang	money	n000	make	yesno
278	0.000	0.778	0.00	0.00	0.00	y
1028	0.180	0.372	0.43	0.43	0.21	y
2259	0.184	0.276	0.06	1.16	0.06	y
191	0.000	0.137	0.00	0.00	0.00	y
191	0.000	0.135	0.00	0.00	0.00	y
54	0.000	0.000	0.00	0.00	0.00	y

Model Evaluation – Spam Filter

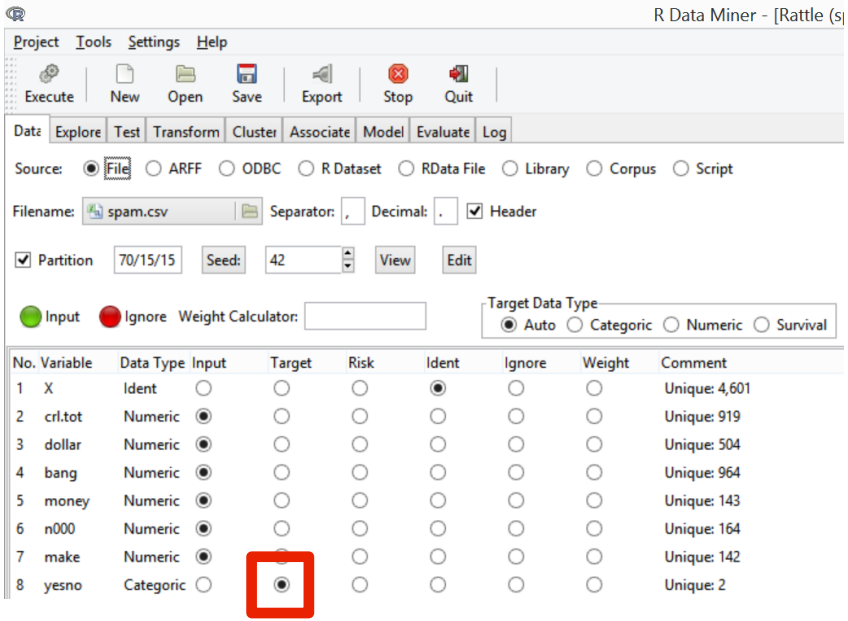
We will build a Spam Filter based on Classification Tree and evaluate its performance based on the following metrics

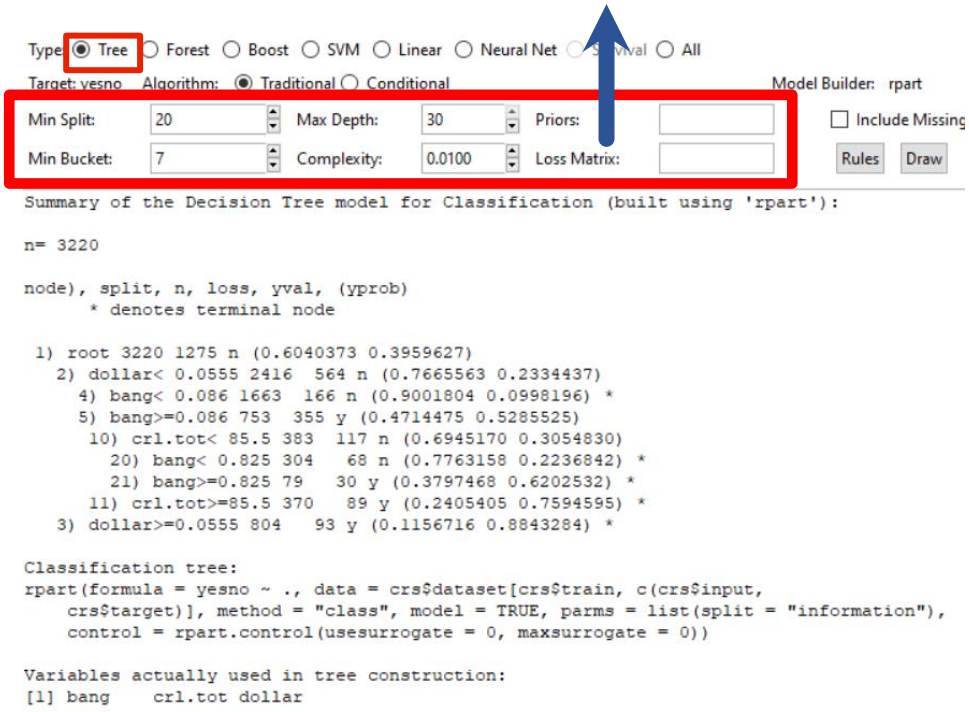
- Error rate
- True and False Positives and Negatives
- Classification: Scoring – to predict the class label
- Precision, Recall, Sensitivity, Specificity
- Confusion matrix
- Risk chart
- Lift chart

Classification Tree

Output slightly changed in R 3.6.3

Default values

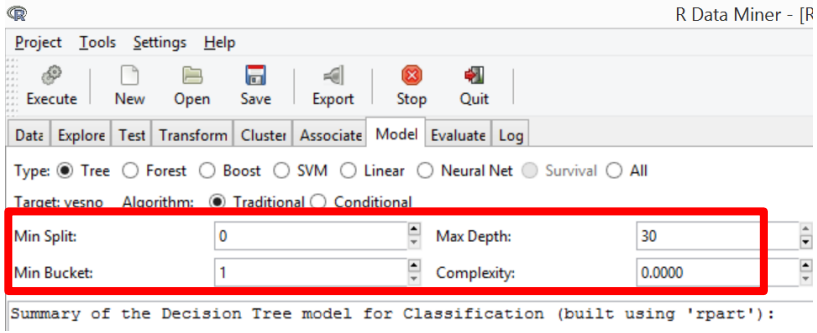




To see the tradeoff between the bias (under fitting) and variance (over fitting), we will first fit the model as complex as possible.

Error Rate

Error rate on training data and validation data for the most complex tree possible in Rattle

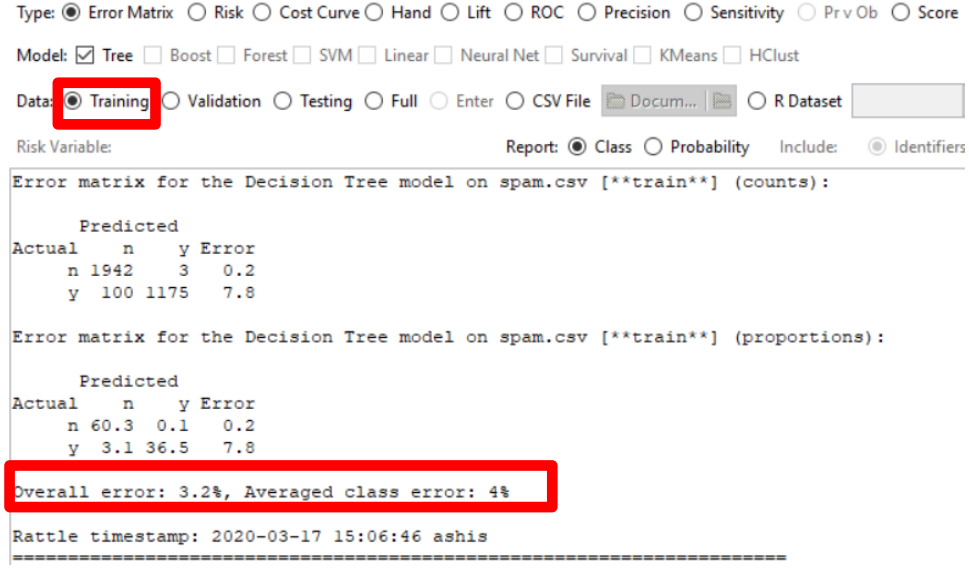


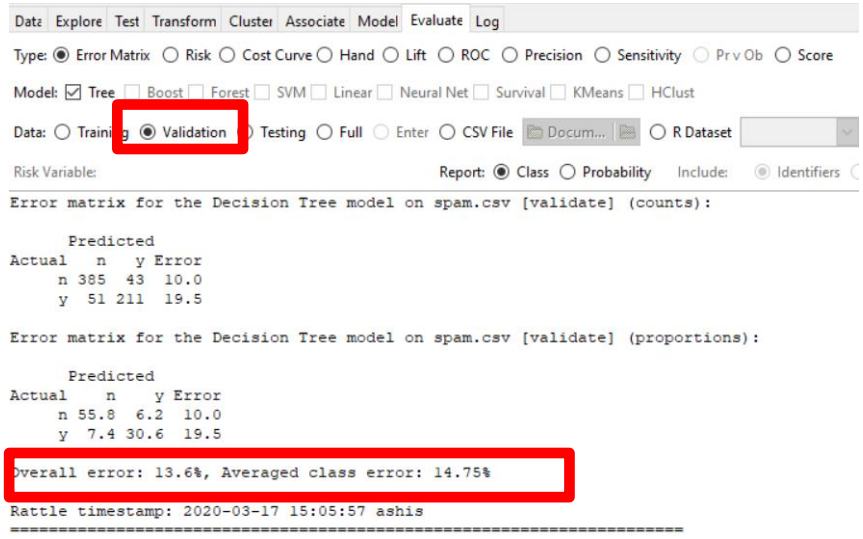
Set “min split”, “min bucket size”, and “complexity parameter” to their minimum values and “depth” to its maximum value in Rattle. *Note that complexity value lower means more complex model*

Source: Rattle GUI / Togaware

Error Rate

Output slightly changed in R 3.6.3

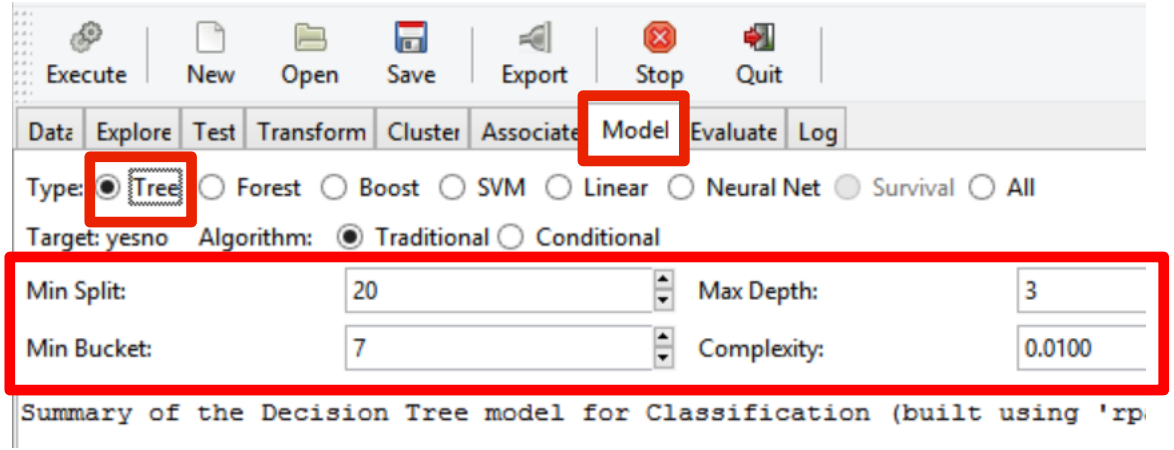




Source: Rattle GUI / Togaware

Error Rate (Default Values except Max Depth = 3)

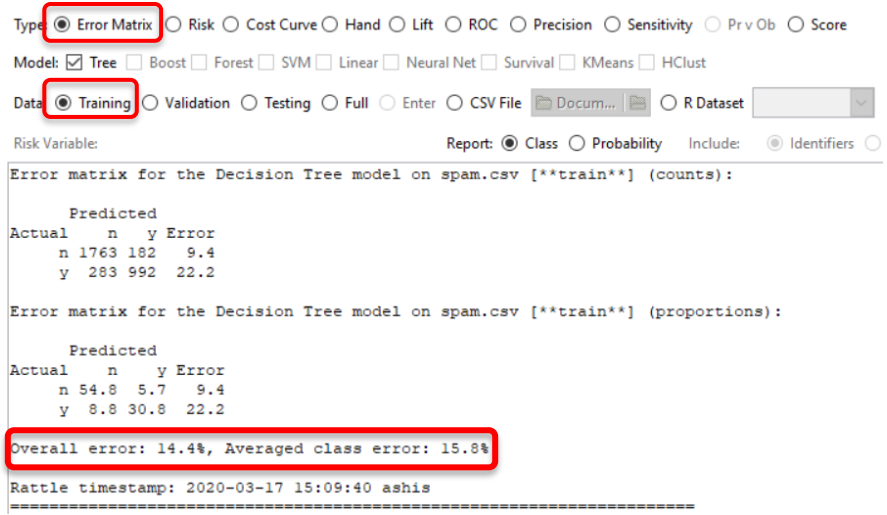
Error rate on training data and validation data for less complex tree (default values in Rattle)

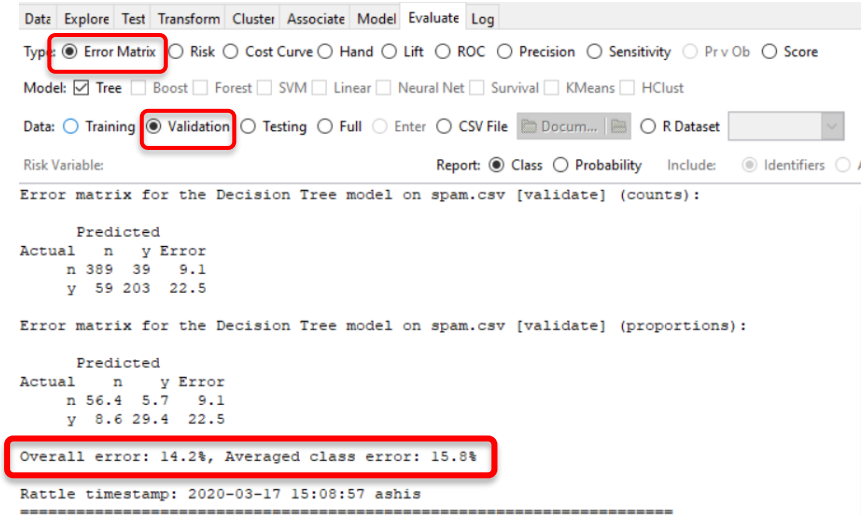


Source: Rattle GUI / Togaware

Error Rate (Default Values)

Output slightly changed in R 3.6.3





Source: Rattle GUI / Togaware

Cutoff for Classification

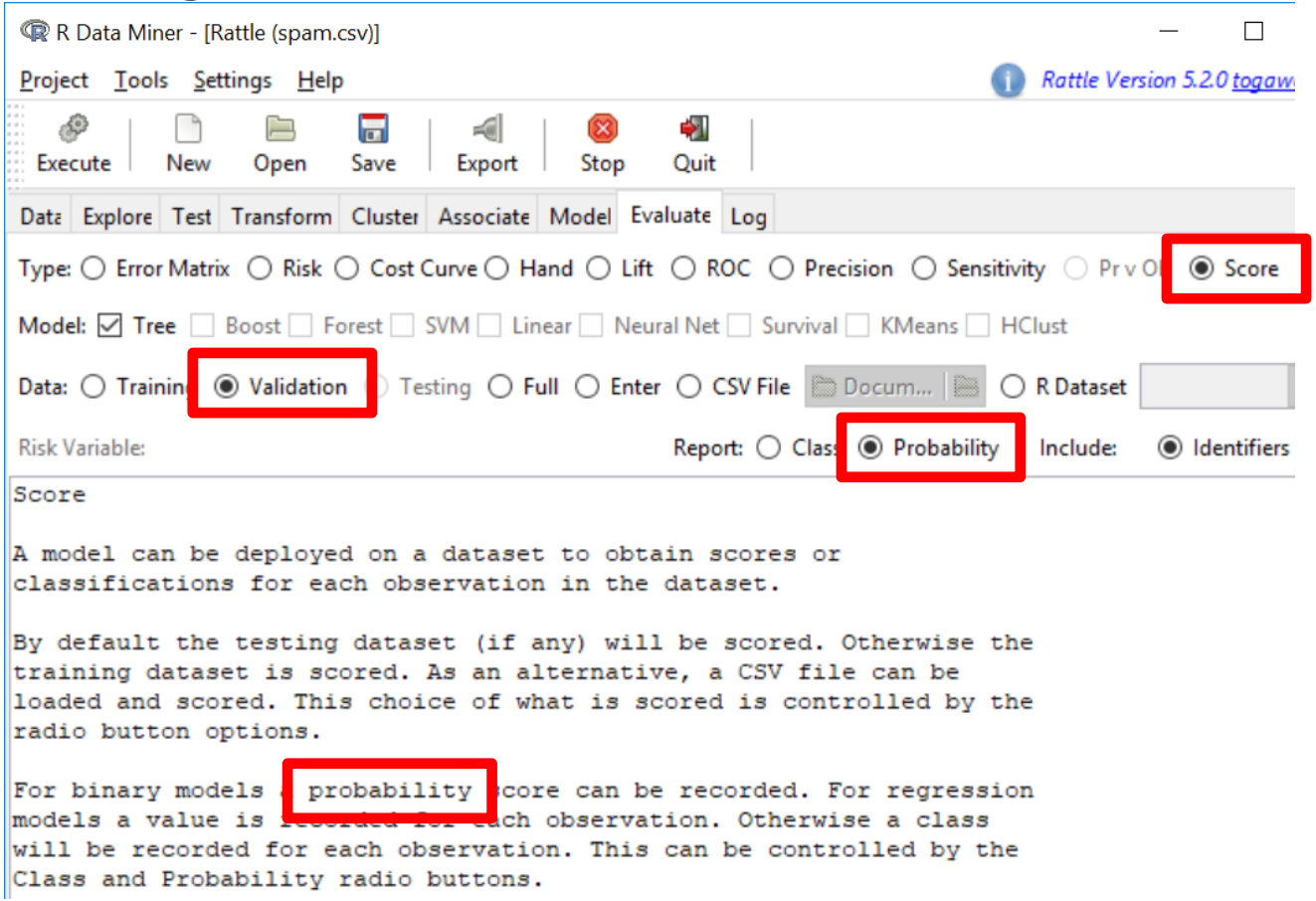
Assign to the class with the highest probability of belonging to that class

Extension to multiple classes by comparing one class against all others

Default probability cutoff is 0.5

To demonstrate used 99:01:00 split. Later will change back

Probability



Source: Rattle GUI / Togaware

Validate Score (new slide)

Output slightly changed in R 3.6.3

	A	B	C
1	YesNo	rpart	Predict
2	y	0.761628	y
3	y	0.761628	y
4	y	0.322761	n
5	y	0.881402	y
6	y	0.881402	y
7	y	0.761628	y
8	y	0.881402	y
9	y	0.881402	y
10	y	0.881402	y
11	y	0.881402	y
12	y	0.881402	y
13	y	0.881402	y
14	y	0.761628	y
15	y	0.881402	y
16	y	0.761628	y
17	y	0.881402	y
18	y	0.881402	y
19	y	0.881402	y
20	y	0.100042	n
21	y	0.881402	y

H	I	J	K	L	M
		Predict			
		0	1	Error	
Actual	0	18	2	0.1	
	1	5	22	0.185185	
	Cutoff	0.5			

spam_validate_score_identsNew.xlsx

CUTOFF = 0.5

Validate Score (new slide)

Output slightly changed in R 3.6.3

	A	B	C
1	YesNo	rpart	Predict
2	y	0.761628	n
3	y	0.761628	n
4	y	0.322761	n
5	y	0.881402	y
6	y	0.881402	y
7	y	0.761628	n
8	y	0.881402	y
9	y	0.881402	y
10	y	0.881402	y
11	y	0.881402	y
12	y	0.881402	y
13	y	0.881402	y

H	I	J	K	L	M
		Predict			
		0	1	Error	
Actual	0	19	1	0.05	
	1	11	16	0.407407	
	Cutoff	0.8			

spam_validate_score_identsNew.xlsx

CUTOFF = 0.8

True and False Negatives

Output slightly changed in R 3.6.3

Training Set

Data | Explore | Test | Transform | Cluster | Associate | Model | Evaluate | Log

Type: (•) Error Matrix ( ) Risk ( ) Cost Curve ( ) Hand ( ) Lift ( ) ROC ( ) Precision ( ) Sensitivity

Model: [✓] Tree [ ] Boost [ ] Forest [ ] SVM [ ] Linear [ ] Neural Net [ ] Survival [ ] KMeans [ ] HClust

Data: (•) Training ( ) Validation ( ) Testing ( ) Full ( ) Enter ( ) CSV File servizn ( ) R Dat

Risk Variable: Report: (•) Class ( ) Probability Include: ( ) Identifiers ( ) A

```
Error matrix for the Decision T                    **] (counts):

      Predicted
Actual    n    y Error
     n 1763  182   9.4
     y  283  992  22.2

Error matrix for the Decision Tree model on spam.csv [**train**] (proportions):

      Predicted
Actual    n    y Error
     n 54.8  5.7   9.4
     y  8.8 30.8  22.2

Overall error: 14.4%, Averaged class error: 15.8%

Rattle timestamp: 2020-03-17 15:17:44 ashis
```

True Negative

True Positive

Validation Set

Data | Explore | Test | Transform | Cluster | Associate | Model | Evaluate | Log

Type: (•) Error Matrix ( ) Risk ( ) Cost Curve ( ) Hand ( ) Lift ( ) ROC ( ) Precision ( ) Sensitivity

Model: [✓] Tree [ ] Boost [ ] Forest [ ] SVM [ ] Linear [ ] Neural Net [ ] Survival [ ] KMeans [ ] HClus

Data: ( ) Training (•) Validation ( ) Testing ( ) Full ( ) Enter ( ) CSV File servizn ( ) R

Risk Variable: Report: (•) Class ( ) Probability In:

```
Error matrix for the Decision Tree model on spam.csv [validate] (counts):

      Predicted
Actual   n   y Error
     n 389  39   9.1
     y  59 203  22.5

Error matrix for the Decision Tree model on spam.csv [validate] (proportions):

      Predicted
Actual    n    y Error
     n 56.4  5.7   9.1
     y  8.6 29.4  22.5

Overall error: 14.2%, Averaged class error: 15.8%

Rattle timestamp: 2020-03-17 15:15:00 ashis
```

Confusion Matrix

Source: Rattle GUI / Togaware

True and False Negatives

True positive - model predicts Yes in agreement with the actual outcome (actual outcome = Yes)

True Negative - model predicts No in agreement with the actual outcome (actual outcome = No)

False Positive - model predicts Yes in **DIS**agreement with the actual outcome (actual outcome = No)

False Negative - model predicts No in **DIS**agreement with the actual outcome(actual outcome = Yes)

Specificity and Sensitivity

Output slightly changed in R 3.6.3

Confusion Matrix Validation Data

Actual	Predicted n	Predicted y	Error
n	389	39	9.1
y	59	203	22.5

Precision = TP/ (TP+FP) = 203/(203+39) = 0.84

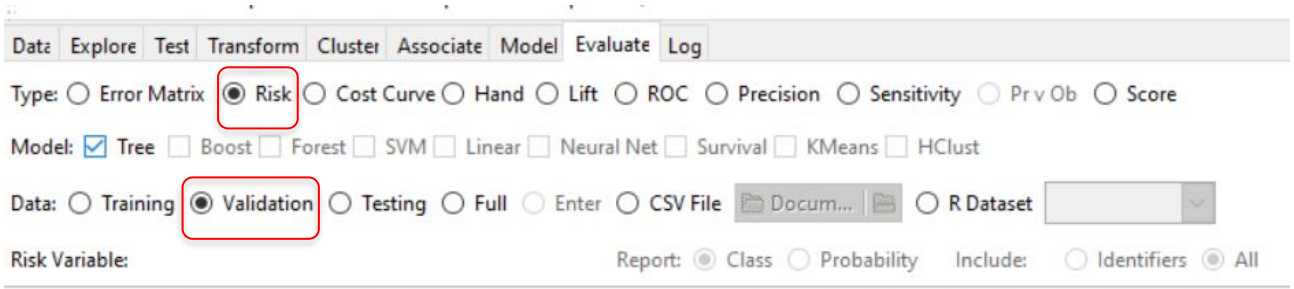
Recall/Sensitivity/True Positive Rate = TP/ (TP+FN) = 203/(203+59) = 0.77

Specificity/True Negative Rate = TN/ (TN+FP) = 389/(389+39) = 0.91

Source: Rattle GUI / Togaware

Risk

Output slightly changed in R 3.6.3




```
Summary Decision Tree model (built using rpart) on spam.csv [validate] by probability cutoffs.

                  Recall  Caseload Precision
0.0998196031269 1.0000000 1.0000000 0.3797101
0.3054830287206 0.8740458 0.4594203 0.7223975
0.7594594594595 0.7748092 0.3507246 0.8388430
0.884328358209  0.5343511 0.2347826 0.8641975
1.0             0.0000000 0.0000000 1.0000000

Rattle timestamp: 2020-03-17 15:19:34 ashis
========================================================================
The area under the Risk and Recall curves for Decision Tree model

Area under the Recall (green) curve: 91% (0.907)


Rattle timestamp: 2020-03-17 15:19:34 ashis
========================================================================
```

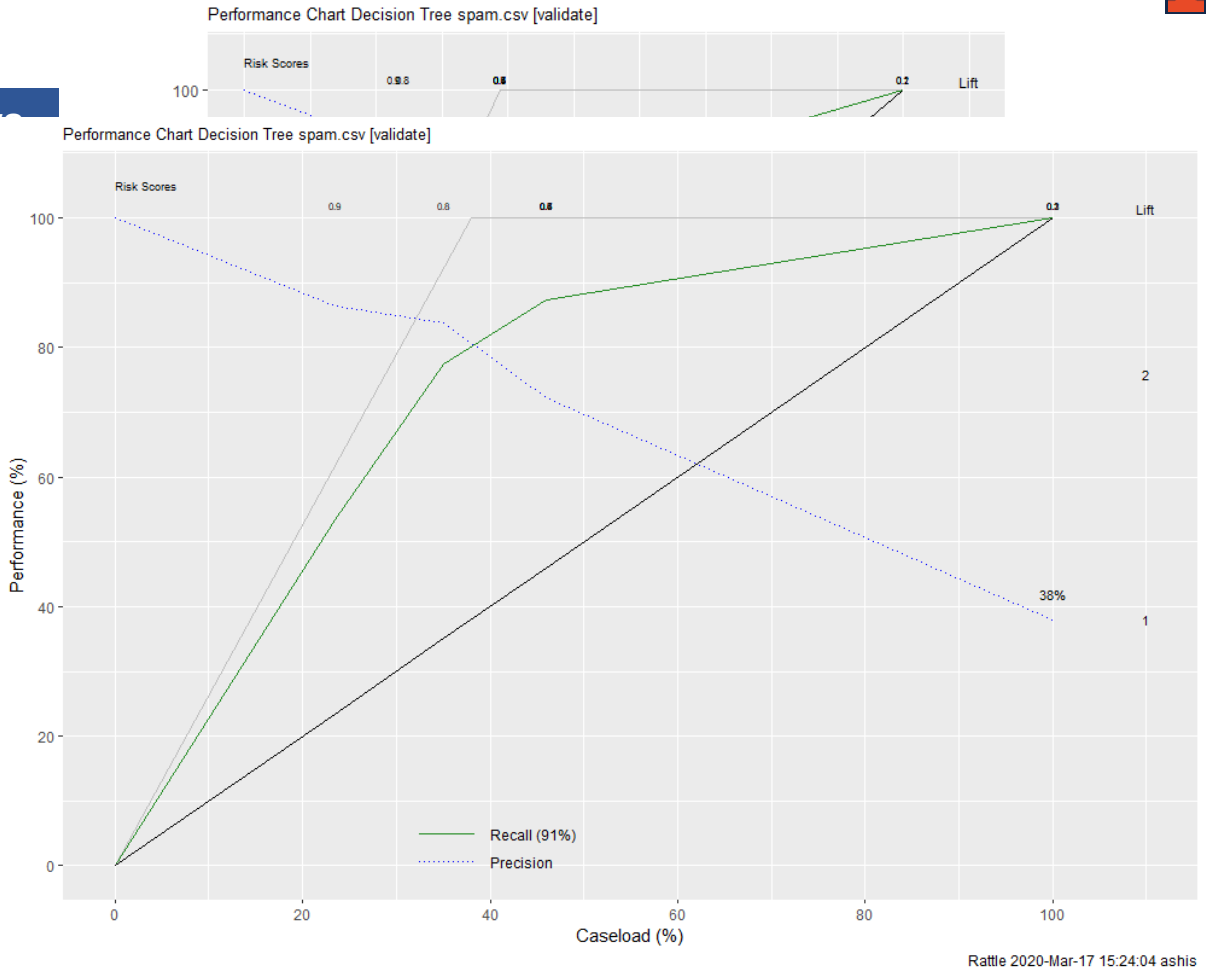
Source: Rattle GUI / Togaware

Risk

Risk Scores/Precision - This shows what percent of our predicted Spams are indeed spams.

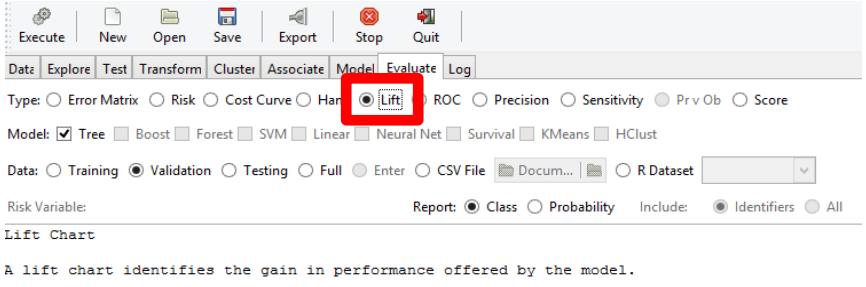
Recall - This shows what percent spams were we able to actually predict as spams.

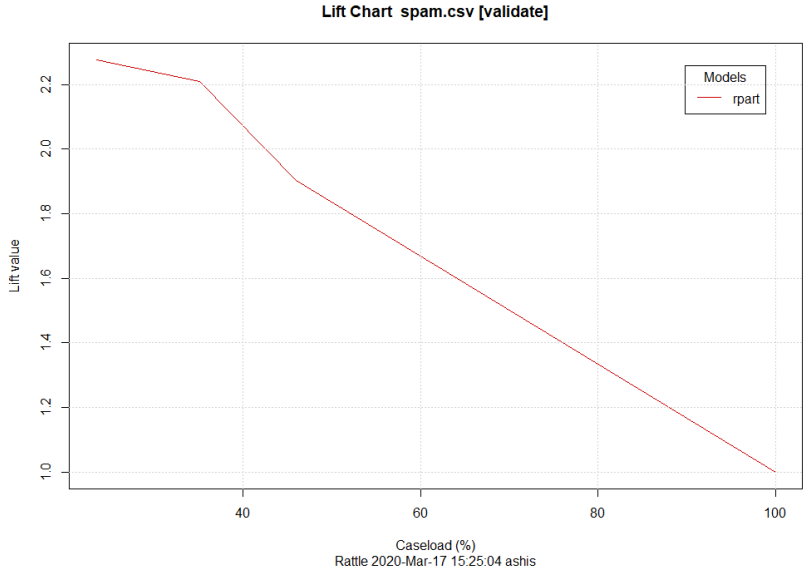
Lift = Risk score / % of actua order of predicted probabilit the proportion of actual "yes



Evaluation – Lift Chart

Output slightly changed in R 3.6.3



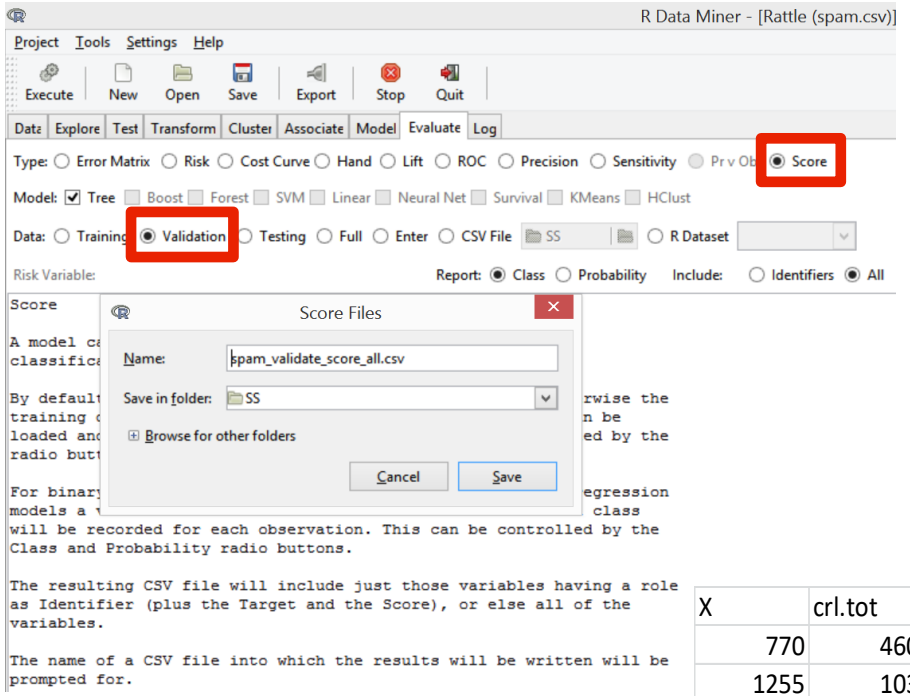


Source: Rattle GUI / Togaware

Lift = Risk score / % of actual "yes" in the overall data.

A lift value in the first 40% (sorted in descending order of predicted probability of yes) on the of data being 2 means that in the 4th decile, the proportion of actual "yes" is 2 times the proportion of "yes" in the overall data.

Predicted Class



Predicted Class: Spam or not

X	crl.tot	dollar	bang	money	n000	make	yesno	rpart
770	460	0.163	0.02	0.05	0.75	0.17	y	y
1255	103	0	1.417	0.71	0	0.71	y	y
3489	44	0	0	0	0	0	n	n
3781	109	0	0	0	0	0	n	n
2538	21	0	0	0	0	0	n	n

Source: Rattle GUI / Togaware

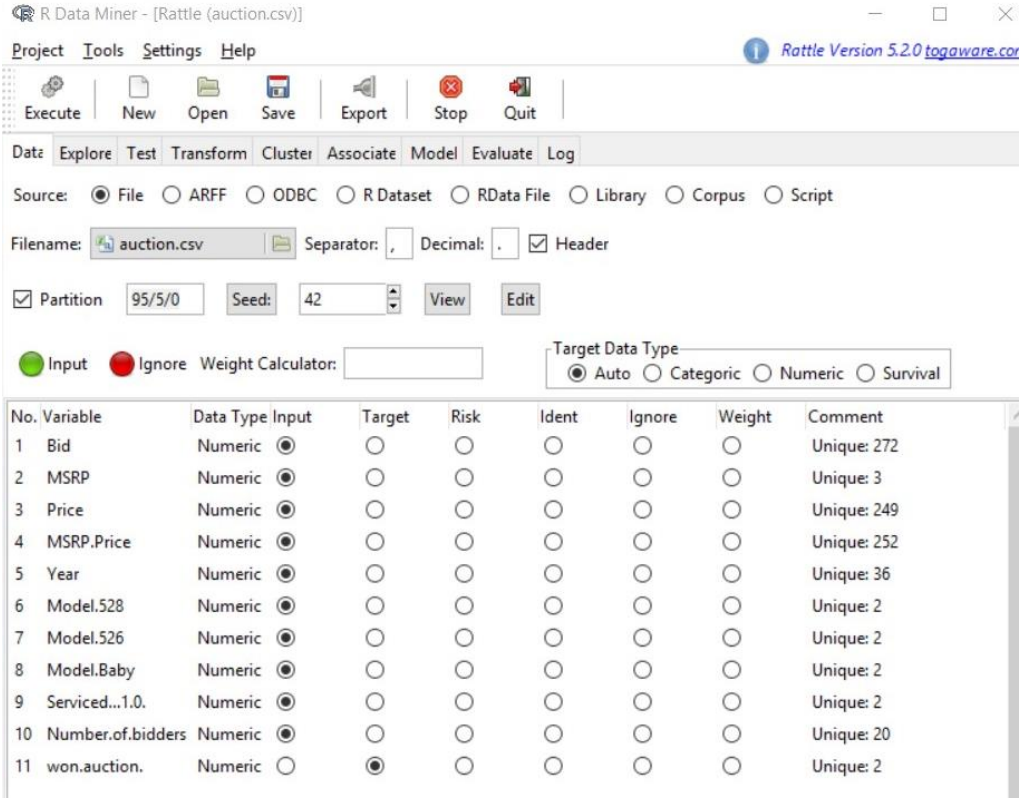
Output different in R 3.6.3

Exercise

Calculate the metrics for the **auction data** using Random Forest to obtain and comment. The table below was produced by analyzing the probability score on validate data in Microsoft Excel. The Rattle steps are as shown in next three slides.

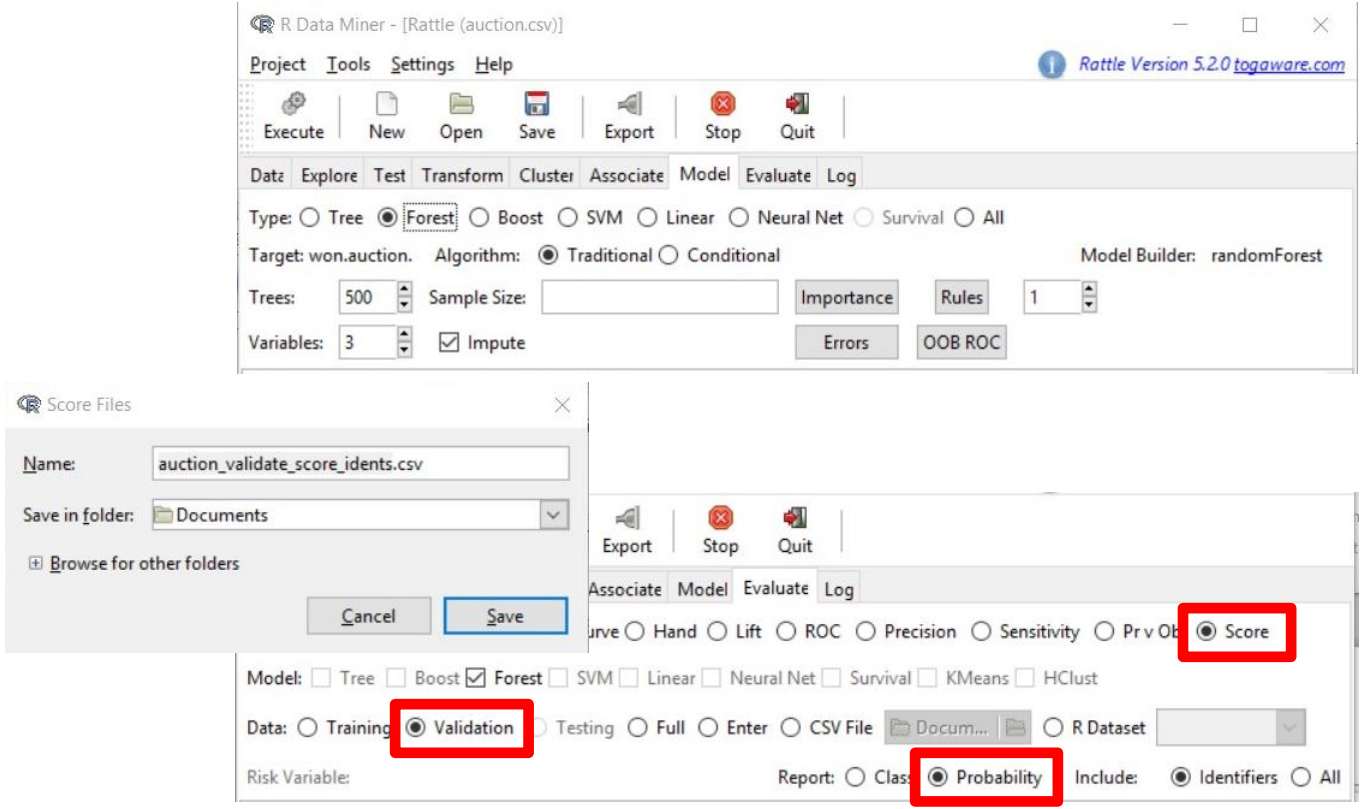
Cutoff	Error	Sensitivity	Specificity	Precision
	0.071429	0.8	1.000	1.000
0.1	0.428571	0.8	0.444	0.444
0.2	0.357143	0.8	0.556	0.500
0.3	0.285714	0.8	0.667	0.571
0.4	0.142857	0.8	0.889	0.800
0.5	0.071429	0.8	1.000	1.000
0.6	0.071429	0.8	1.000	1.000
0.7	0.214286	0.4	1.000	1.000

Exercise - Run the RF Method on Auction Data



Source: Rattle GUI / Togaware

Exercise - Extract the Score



Source: Rattle GUI / Togaware

Exercise - The Scores

Output different in R 3.6.3

	A	B	C	D	E	F	G	H	I	J	K
1	won.auction.	rf	Predict	0,0	0,1	1,0	1,1			Predict	
2	0	0.172	0	1	0	0	0			lost	won
3	0	0.138	0	1	0	0	0		lost	9	3
4	0	0.132	0	1	0	0	0	Actual	won	0	2
5	0	0.366	1	0	1	0	0				
6	0	0.89	1	0	1	0	0		Cutoff = 0.2		
7	0	0.25	1	0	1	0	0				
8	0	0.092	0	1	0	0	0				
9	0	0.024	0	1	0	0	0				
10	0	0.05	0	1	0	0	0				
11	0	0.12	0	1	0	0	0				
12	1	0.418	1	0	0	0	1			Predict	
13	0	0.064	0	1	0	0	0			lost	won
14	0	0.102	0	1	0	0	0		lost	11	1
15	1	0.24	1	0	0	0	1	Actual	won	2	0
16				9	3	0	2				
17									Cutoff = 0.5		
18	Cutoff	0.2									
19											

Making Recommendations

Data Organization

MULTIVARIATE – table of features

Traditional databases (more later)

BASKET – an **UNWEIGHTED SET** of items

Market Basket, Keyword Sets, Tag Sets,…

BAG – a **WEIGHTED SET** of items

Bag-of-Words, Bag-of-Visual-Words,…

SEQUENCE – of **SYMBOLIC** tokens

Word, Genes, Phonemes,…

SERIES – of **NUMERIC** values

Time Series, Seismographic, EEG,

“Item-Set” – Basket Data

Retail – Market Basket of a customer

Movies – Sets of movies a user likes

Flickr – tags describing **images**

YouTube – tags describing **videos**

“Item-Set” – Basket Data

AdWords – tags describing **advertisements**

IMDB – tags describing **movies**

Keywords – tags describing **scientific publications**

The definition of a basket goes beyond the “Market Basket” – depends on application!

Item-set Data

A Universe (Dictionary) of ALL Items:

Products | Keywords | Movies | …

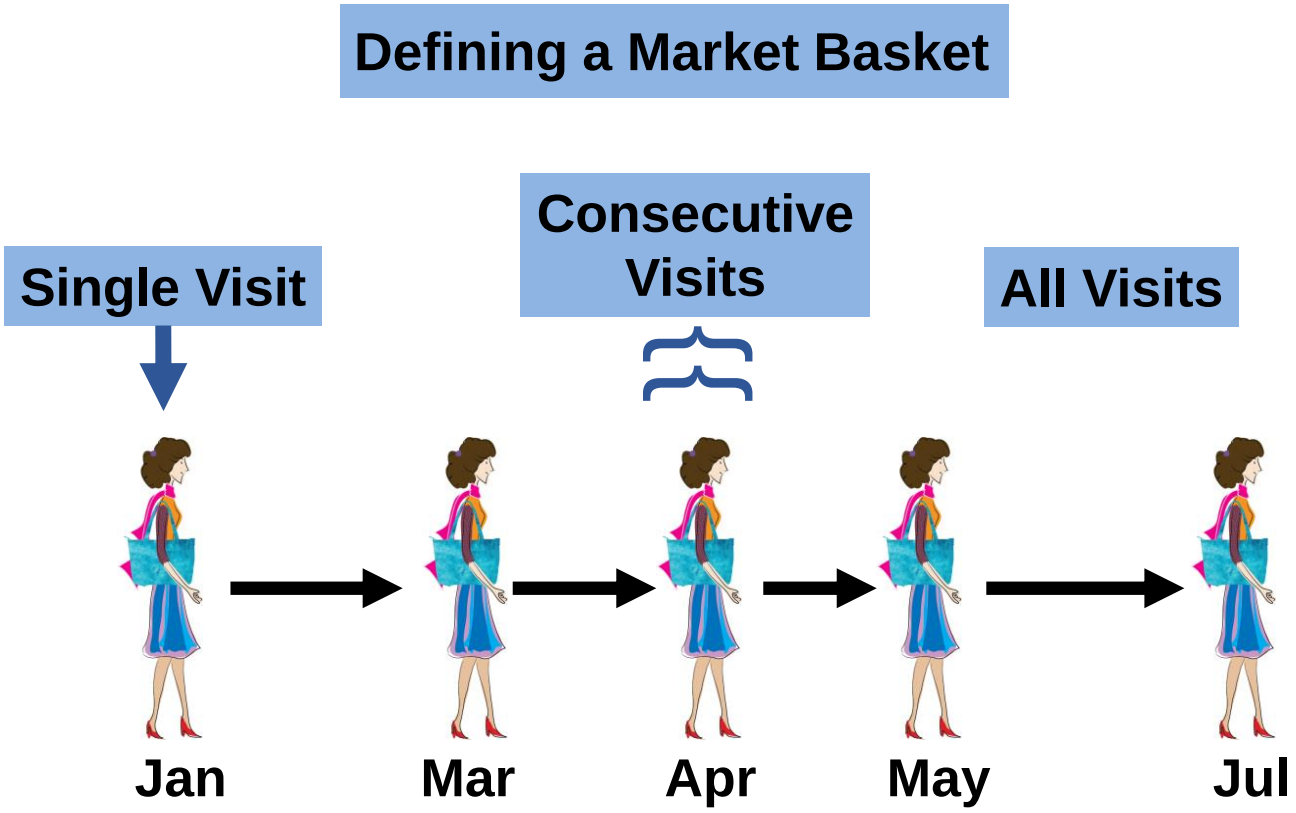
$$\underset{?}{\mathbf{V}} = \{ v_1, v_2, ..., v_N \}$$

A Collection of Item-sets:

$$\underset{?}{\mathbf{X}} = \{ \mathbf{x}_n \grave{\imath} \ \mathbf{V} \}_{n=1}^{N}$$

a, b, c, d
a, b, d, e
b, e, f
a, e, f, g
b, d, e, f
a, f, g
b, c, e, f

Market Basket



Frequent Item-Set Mining

“Frequent itemset mining has been one of the early algorithms that almost gave birth to the field of “data mining.” It was the first breakthrough of its kind in mining such itemset data and since then, there have been a number of improvements in smart data structures to store the candidate and frequent itemsets to make it faster and more scalable. It has also been applied to areas beyond retail data mining for which it was originally invented. It has been used to discover “higher order features” of type “sets of items” in various domains including computer vision.

p. 502, Chapter 15 Unsupervised Machine Learning, Shaliesh Kumar in Pochiraju and Seshadri, Essentials of Business Analytics

Apriori Algorithm

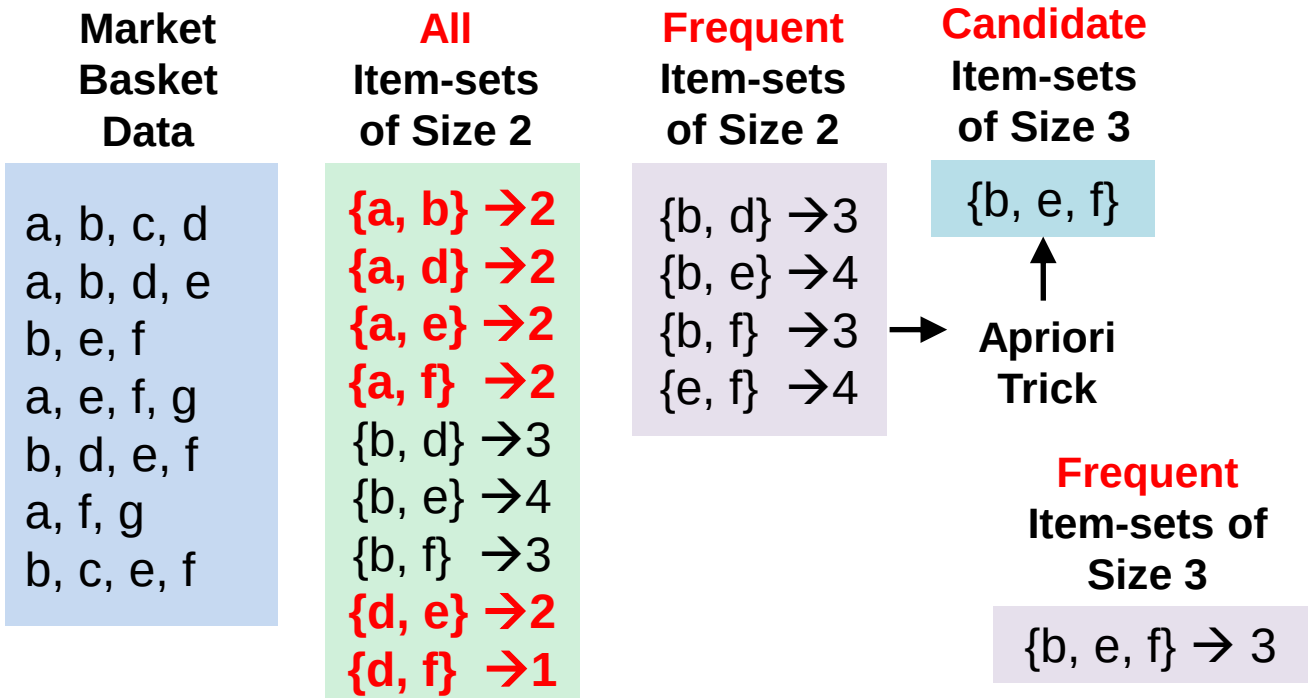
Support Threshold = 3

Market Basket Data	All Item-sets of Size 1	Frequent Item-sets of Size 1	Candidate Item-sets of Size 2	All Item-sets of Size 2
a, b, c, d	{a} → 4	{a} → 4	{a, b}	**{a, b} →2**
a, b, d, e	{b} → 5	{b} → 5	{a, d}	**{a, d} →2**
b, e, f	**{c} → 2**	{d} → 3	{a, e}	**{a, e} →2**
a, e, f, g	{d} → 3	{e} → 5	{a, f}	**{a, f} →2**
b, d, e, f	{e} → 5	{f} → 5	{b, d}	{b, d} →3
a, f, g	{f} → 5		{b, e}	{b, e} →4
b, c, e, f	**{g} → 2**		{b, f}	{b, f} →3
			{d, e}	**{d, e} →2**
			{d, f}	**{d, f} →1**
			{e, f}	{e, f} →4

Apriori Trick →

Support({a}) = 4
Support({g}) = 2

Apriori Algorithm



Frequent Itemsets → Association Rules

Market Basket Data

a, b, c, d
a, b, d, e
b, e, f
a, e, f, g
b, d, e, f
a, f, g
b, c, e, f

Frequent Item-sets of Size 1

{a} → 4
{b} → 5
{d} → 3
{e} → 5
{f} → 5

Frequent Item-sets of Size 2

{b, d} →3
{b, e} →4
{b, f} →3
{e, f} →4

Association Rules

{b} → {d}
{d} → {b}
{b} → {e}
{e} → {b}
{b} → {f}
{f} → {b}
{e} → {f}
{f} → {e}

$$Confidence\left(\{b\} \rightarrow \{f\}\right) \equiv P\left(\{f\} \middle| \{b\}\right) = \frac{Support\left(\{b, f\}\right)}{Support\left(\{b\}\right)} = \frac{3}{5}$$

Frequent Itemsets → Association Rules

Market Basket Data

a, b, c, d
a, b, d, e
b, e, f
a, e, f, g
b, d, e, f
a, f, g
b, c, e, f

Frequent Item-sets of Size 2

{b, d} →3
{b, e} →4
{b, f} →3
{e, f} →4

Frequent Item-sets of Size 3

{b, e, f} → 3

Association Rules

{b, e} → {f}
{b, f} → {e}
{e, f} → {b}

{b} → {e, f}
{e} → {b, f}
{f} → {b, e}

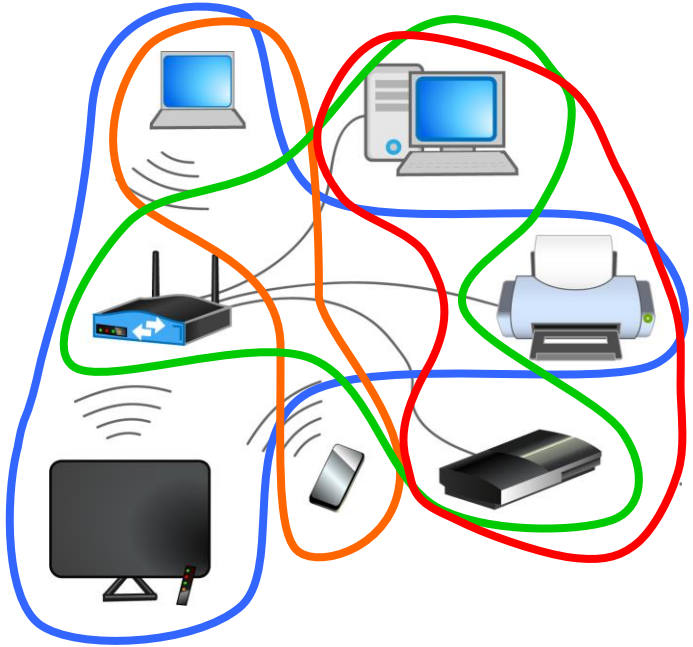
$$Conf\left(\{b,f\} \rightarrow \{e\}\right) = \frac{Support\left(\{b,e,f\}\right)}{Support\left(\{b,f\}\right)} = ??$$

$$Conf\left(\{b\} \rightarrow \{e,f\}\right) = ??$$

Retail Data – A “Crazy Haystack”

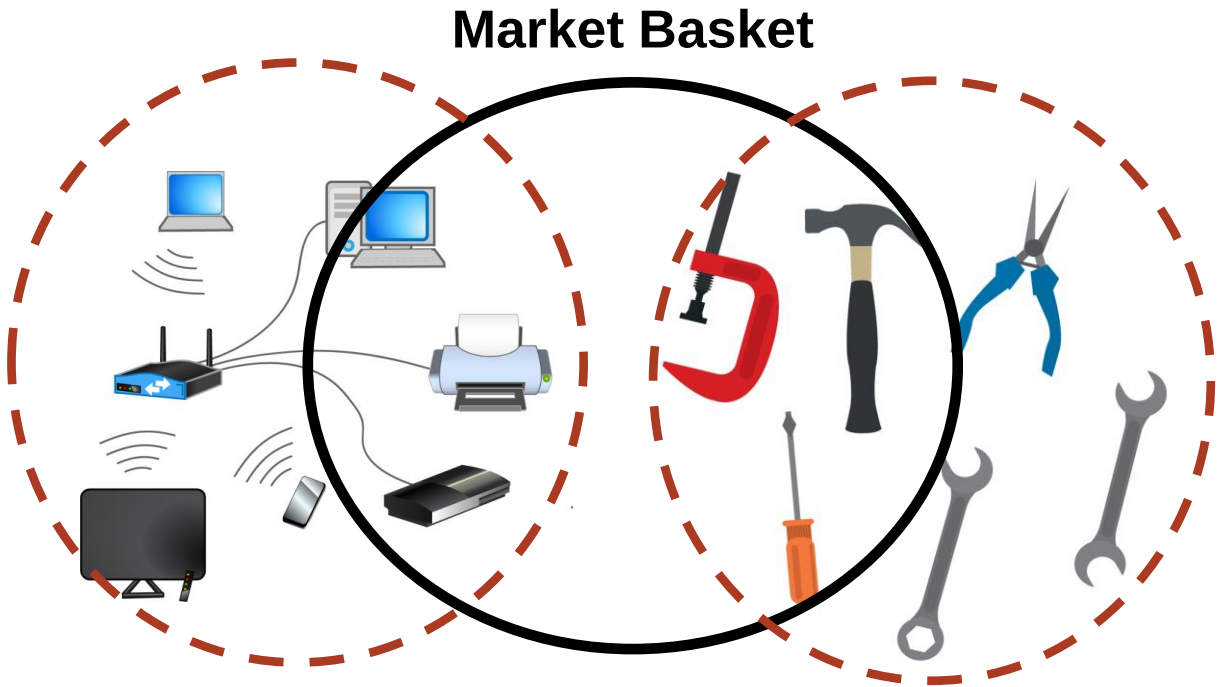
Few buy a complete “logical” item-set in same basket

- Already have other products
- Buy them from another retailer
- Buy them at a different time
- Got them as gifts
- Etc



It’s a **projection** of **latent customer** intentions

It Gets Even Crazier



It’s a **mixture** of **projections** of **latent** intentions

How to Recommend Other Topics for Further Study

Advanced regression models

SVM

Neural networks

Deep Learning

Philosophy

Association Rule Mining

This algorithm can take data in two formats: Basket (each row is a transaction, where each item purchased is given as a list) and Dataframe (where each row is a transaction and each column represents an item that was bought or not)

Association Rule Mining

For Basket format, we will use the DVD_Transaction data from Rattle, and for Dataframe format we will use a subset of Groceries data R Library.

DVD Transaction

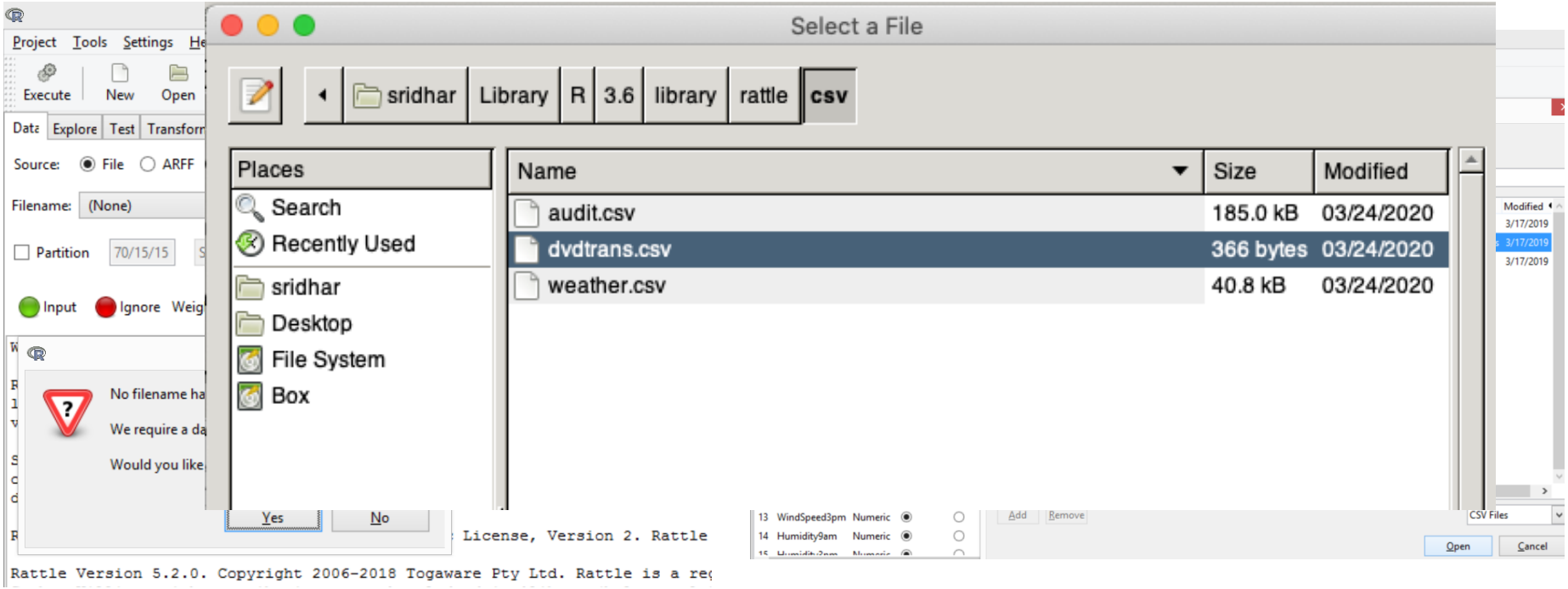
Predicting which movie is customer likely to order based on his movie watching history

To get the dvd_trans file into Rattle, follow the following steps:

- Load rattle() from R
- Without selecting any data file, press *Execute*
- The rattle would ask permission to load default weather.csv, say *Yes*
- Then click on the *File* button and you will see dvd_trans.csv, select it

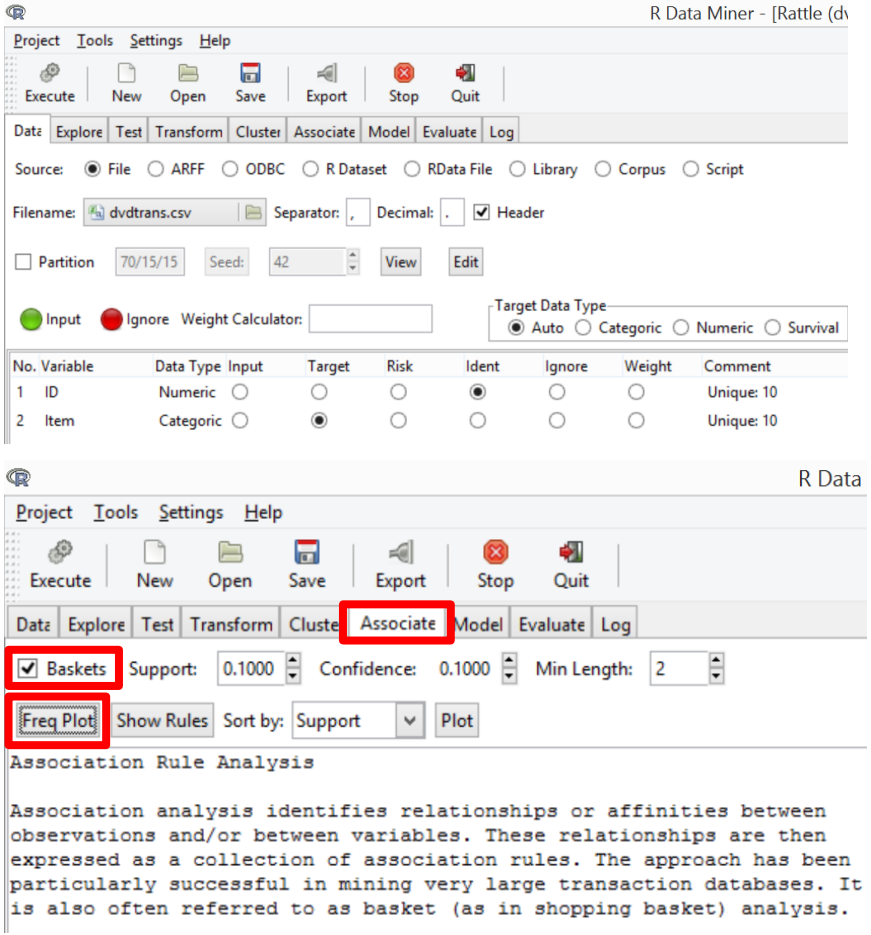
DVD Transaction

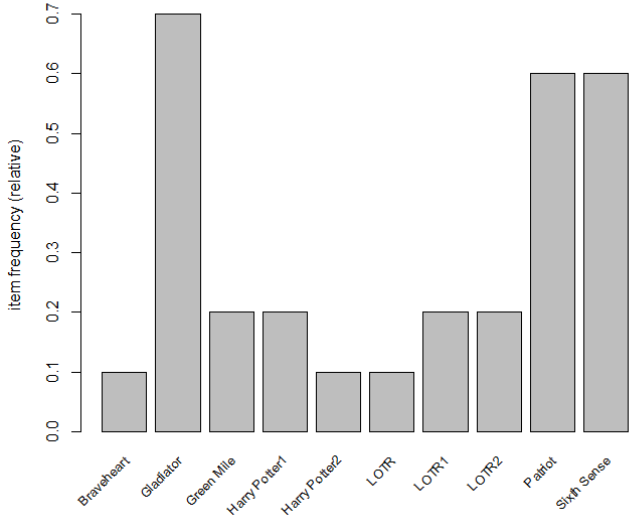
The file location is machine dependent



Source: Rattle GUI / Togaware

DVD_Transactions



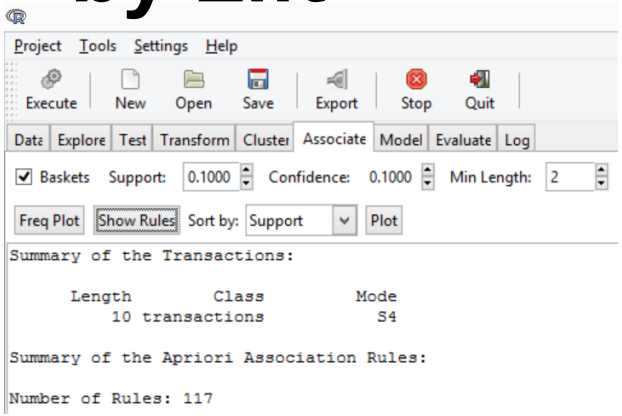


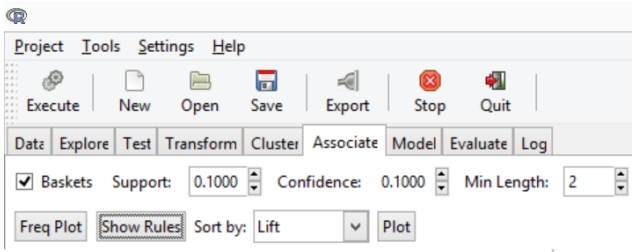
Relative frequency among the items

Source: Rattle GUI / Togaware

Rules: Sorted by Support and Sorted by Lift

If sort does not work copy to excel and sort





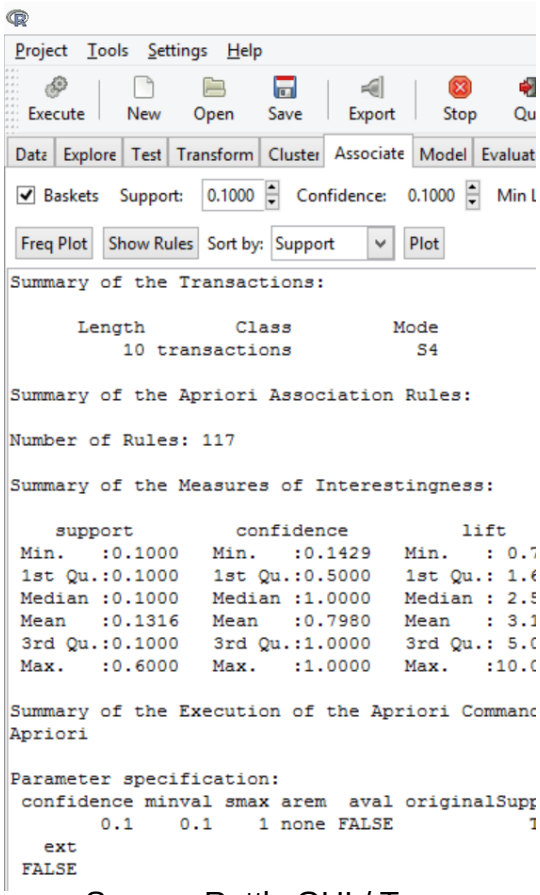
	lhs		rhs	support	confidence	lift	count
[1]	{Patriot}	=>	{Gladiator}	0.6	1.0000000	1.4285714	6
[2]	{Gladiator}	=>	{Patriot}	0.6	0.8571429	1.4285714	6
[3]	{Sixth Sense}	=>	{Gladiator}	0.5	0.8333333	1.1904762	5
[4]	{Gladiator}	=>	{Sixth Sense}	0.5	0.7142857	1.1904762	5
[5]	{Patriot}	=>	{Sixth Sense}	0.4	0.6666667	1.1111111	4

The rule {Gladiator, Green Mile} → {LOTR} has lift =10, which means probability of watching {LOTR} is 10 times, if we know that the person has watched {Gladiator, Green Mile} , the probability of watching {LOTR} without any information about his watching history.

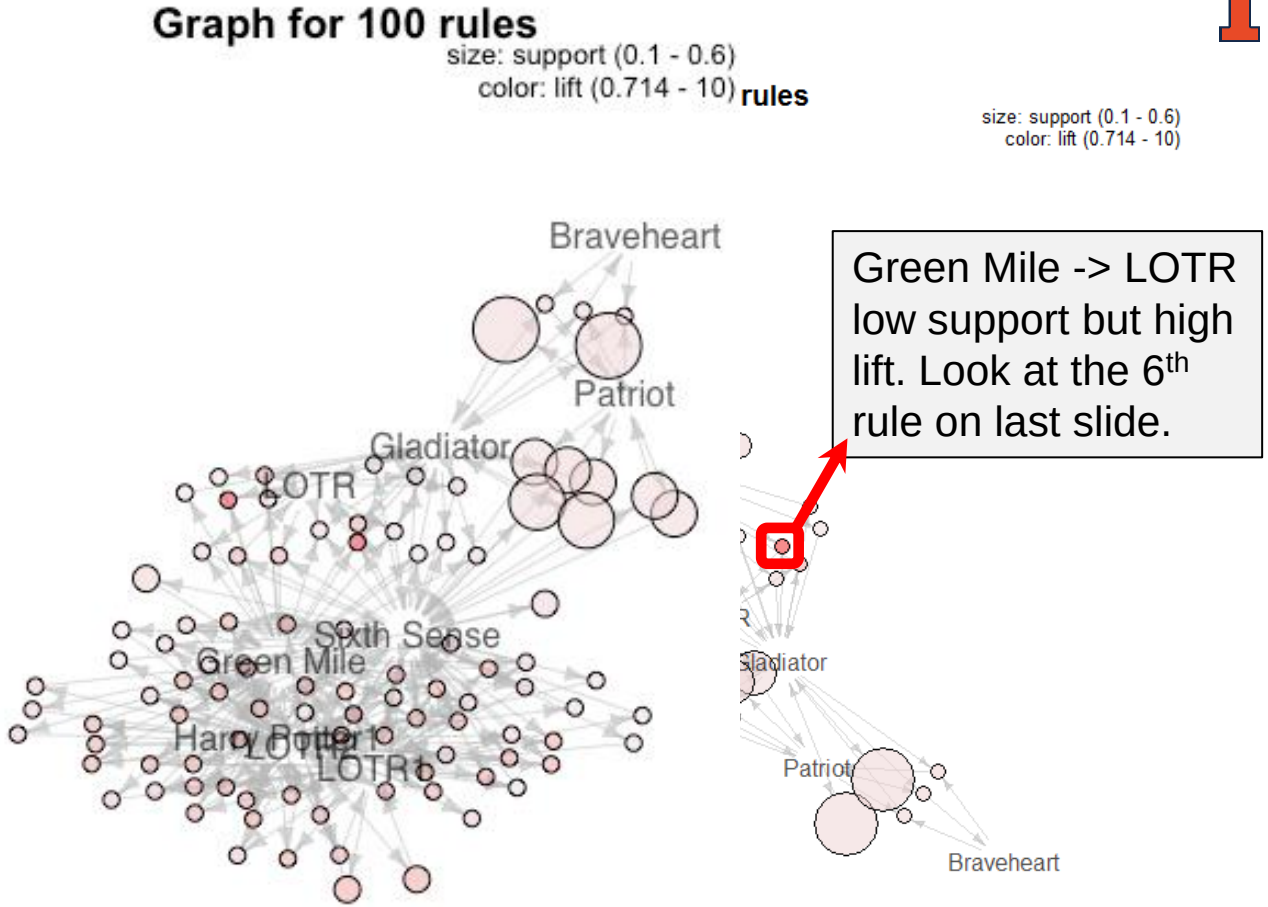
	lhs		rhs	support	confidence	lift	count
[1]	{Gladiator,Green Mile}	=>	{LOTR}	0.1	1.0000000	10.0000000	
[2]	{Gladiator,Green Mile,Sixth Sense}	=>	{LOTR}	0.1	1.0000000	10.0000000	
[3]	{Harry Potter2}	=>	{Harry Potter1}	0.1	1.0000000	5.0000000	
[4]	{Harry Potter1}	=>	{Harry Potter2}	0.1	0.5000000	5.0000000	
[5]	{LOTR}	=>	{Green Mile}	0.1	1.0000000	5.0000000	
[6]	{Green Mile}	=>	{LOTR}	0.1	0.5000000	5.0000000	

Source: Rattle GUI / Togaware

Plots



Source: Rattle GUI / Togaware



Green Mile -> LOTR low support but high lift. Look at the 6th rule on last slide.

Exercise

The library file is bad. We use a subset.

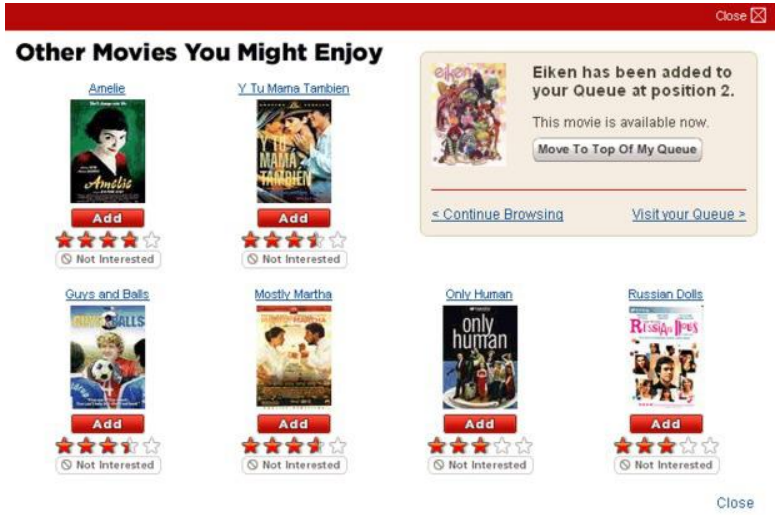
The subset file is in data under lecture 8. Remember to convert to categorical

Grocery data as exercise.

Exercise – Use the groceries subset data in R Library (Groceries:arules:Groceries Data Set) to mine association rules and comment upon them.

Recommender System

Predict ratings and create personalized recommendations for products like books, songs or movies



https://d4datascience.wordpress.com/2016/07/22/recommender-systems-101/

https://pixabay.com/photos/shelf-toys-shop-store-childhood-830421/

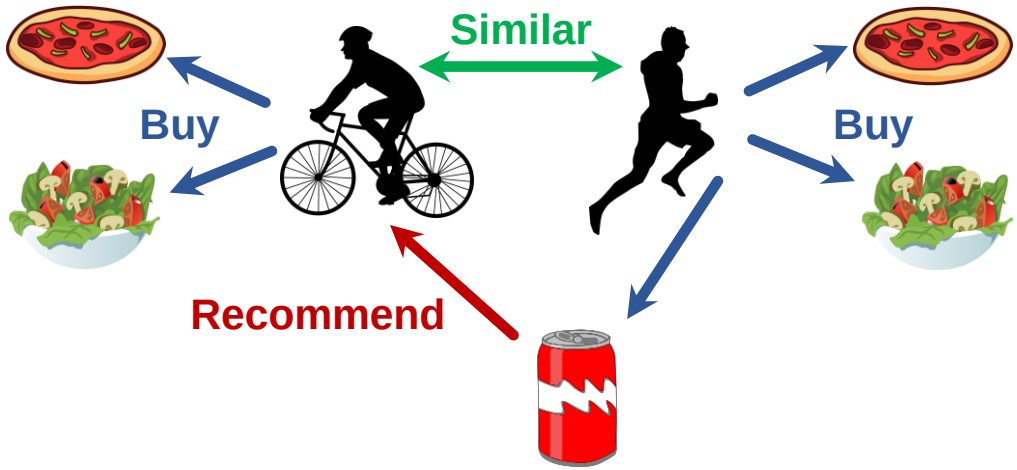
Types of Recommender Systems

Collaborative Filtering

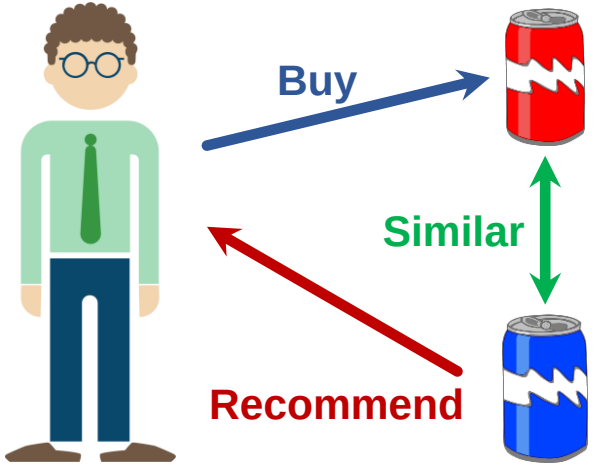
Content-Based Filtering

Hybrid Recommendation Systems – combine above two

Collaborative Filtering



Content-Based Filtering



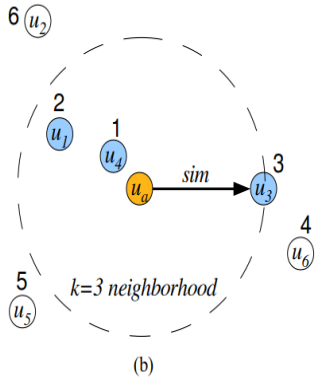
Type of Collaborative Filtering

User-based Collaborative Filtering

We want to predict the ratings for Items i_1 , i_2 , i_5 , and i_7 for the User u_a

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	?	4.0	4.0	2.0	1.0	2.0	?	?
u_2	3.0	?	?	?	5.0	1.0	?	?
u_3	3.0	?	?	3.0	2.0	2.0	?	3.0
u_4	4.0	?	?	2.0	1.0	1.0	2.0	4.0
u_5	1.0	1.0	?	?	?	?	?	1.0
u_6	?	1.0	?	?	1.0	1.0	?	1.0
u_a	?	?	4.0	3.0	?	1.0	?	5.0
$\hat{r}_a$	3.5	4.0			1.3		2.0	

(a)



(b)

Based on the rating provided by ua on items i_3 , i_4 , i_6 , and i_8, *we find other users in the neighborhood of U_a created on the basis of ratings on i_3 , i_4 , i_6 , and i_8. For k=3, The users u_1 , u_4, and u_3 are in the neighborhood. Now, the predicted ratings for u_a for each item is the average rating by users in the neighborhood (the avg. could also be weighted by the others users distance from u_a)*

Hahsler, Michael. *recommenderlab: A framework for developing and testing recommendation algorithms*. 2015.

Type of Collaborative Filtering

Item-based Collaborative Filtering

We want to predict the ratings for Items i_2, i_3 , i_4 , i_6 and i_7 for the User u_a

S	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	$k=3$ $\hat{r}_a$
i_1	-	0.1	0	**0.3**	**0.2**	**0.4**	0	0.1	-
i_2	0.1	-	**0.8**	**0.9**	0	**0.2**	0.1	0	0.0
i_3	0	**0.8**	-	0	**0.4**	0.1	0.3	**0.5**	4.6
i_4	**0.3**	**0.9**	0	-	0	0.1	0	**0.2**	3.2
i_5	**0.2**	0	**0.4**	0	-	0.1	**0.2**	0.1	-
i_6	**0.4**	**0.2**	0.1	**0.3**	0.1	-	0	0.1	2.0
i_7	0	**0.1**	**0.3**	0	**0.2**	0	-	0	4.0
i_8	**0.1**	0	**0.5**	**0.2**	0.1	0.1	0	-	-
u_a	2	?	?	?	4	?	?	5	

We create an item-to-item similarity matrix using any similarity measure (Pearson correlation and Cosine similarity). Choose a "k" and k (=3) largest entries are stored per row (these entries are marked using bold face) In the given matrix – the three items most similar to i_3 are i_2, i_5, and i_8. I_2 can't help in prediction as u_a has not rated it. Thus, we use the ratings provided by the user u_a on items i_5 and i_8. We take weighted average of u_a's ratings on i_5 and i_8 to predict his rating on i_3 . Where, weights are the distance between i_3 and i_5 (0.4) and i_3 and i_8(0.5).

*U_a's rating for i_3 = (0.4*4 + 0.5*5)/(0.4+0.5)*

Hahsler, Michael. *recommenderlab: A framework for developing and testing recommendation algorithms*. 2015.

Overall Idea in the R Script

100 jokes evaluated by 5000 users

Split the data into a train set of 90% and 10% of test(holdout) set

Evaluation Scheme - We need to define an evaluation scheme to make predictions. This evaluation scheme creates train, test data internally for latter usage – we label this as eval

Good rating is the cutoff value of rating above which we would predict the user to like the joke

Once eval created:

For the test set 15 jokes will be given to the recommender algorithm and the other jokes will # be held out for computing the error.

The 15 jokes are used to find users from the train set who are similar to the users (items) from the test set for whom we are evaluating our predictions.

Afterwards you may input your evaluation for fifteen jokes and predicted values for the rest of the jokes.

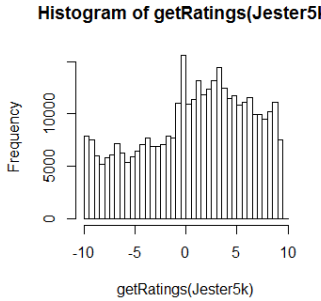
R Script

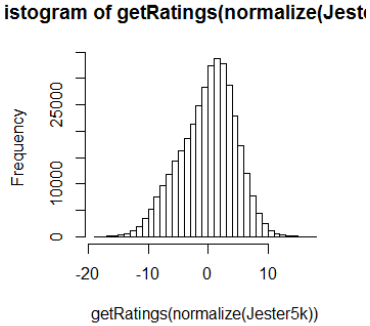
```
install.packages("pacman") # Install the package for managing other packages
library(pacman) # load the "package manage" package in the current session
p_load("recommenderlab") # install load the reqired package(s)

data(Jester5k) # load the jokes data from the recommenderlab library

Jester5k # see the datatype
# 5000 x 100 rating matrix of class ‘realRatingMatrix’ with 362106 ratings.

set.seed(111) # for reproducability of the results

# We can get all the ratings from this‘realRatingMatrix’ Jester5k through the command "getRatings"

# Inspecting the max and min in the data
# Check max rating by any user for any joke
max(getRatings(Jester5k)) #  9.9
# Check max rating by any user for any joke
min(getRatings(Jester5k)) # -9.95
```

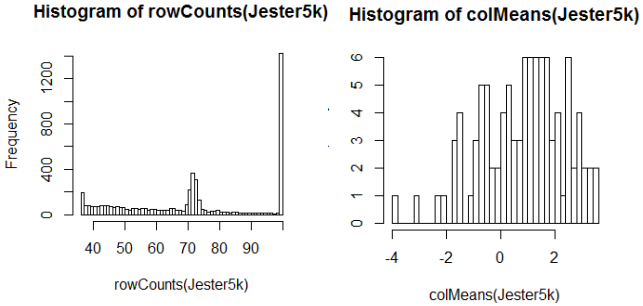
Eigentaste: A Constant Time Collaborative Filtering Algorithm. Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Information Retrieval, 4(2), 133-151. July 2001.

R Script – Continued

```
###----- Inspecting a user's rating pattern -----##
# Check the ratings by any given user
rowCounts(Jester5k[1,]) # u2841 - 81  -- > User u2841 has rated 81 jokes

as(Jester5k[1,], "list") # to see the jokes rated by user u2841
# u2841's rating for joke 1 is "7.91"

rowMeans(Jester5k[1,]) # Average of all rating for this user is 3.855185

############ Inspecting Data Summary ######
# If we use the ratings as provided by the user, the model may have row bias, which is an individual's
# tendency to rate every joke very high or very low
# To remove the row bias, we can normalize the ratings - There are many options for normalization
# such as row centering (difference from the row mean) or Z-score normalization

# Below we show the distribution of the non-normalized and normalized ratings
hist(getRatings(Jester5k), breaks=50) # top figure
hist(getRatings(normalize(Jester5k)), breaks=50) # Normalized 2nd from top figure

hist(rowCounts(Jester5k), breaks = 50) # distribution of # of jokes rated by a user (bottom left)

hist(colMeans(Jester5k), breaks = 50) # distribution of average ratings per joke(bottom right)
```







R Script – Continued

Output different in R 3.6.3

```
# Evaluation Scheme - We need to define an evaluation scheme to make predictions. This evaluation scheme creates train, test data internally for
# later usage
eval <- evaluationScheme(Jester5k[1:2000], method="split", train=0.9, given=15, goodRating=5) # we will use only 2000 users for this exercise out of which  1800 will be in
train set

# Split the data into a train set of 90% and 10% of test(holdout) set
# For the test set 15 jokes will be given to the recommender algorithm and the other jokes will
# be held out for computing the error. The 15 jokes are used to find users from the train set who are similar
# to the users from the test set for whom we are evaluating our predictions.
# Based on the similar users we find ratings for the remaining jokes for the test users.
# Good rating is the cutoff value of rating above which we would predict the user to like the joke

eval # This is a scheme that we will use to train the model and evaluate the model's performance
################################ Building the Recommender System ###################################
# A recommender is created using the creator function Recommender()
# Here "known" are the 15 jokes rating we had given in the test to find similar users from the train set

r_ubcf <- Recommender(getData(eval, "train"), "UBCF") # Recommendation Model generated by user-based collaboration filtering method
pred_ubcf <- predict(r_ubcf, getData(eval, "known"), type="ratingMatrix") # predictions based on UBCF | Predict returns predicted ratings on test data
as(pred_ubcf, "matrix")[1:10,1:10] # Check the predicted ratings of 10 users for first 10 jokes.

r_ibcf <- Recommender(getData(eval, "train"), "IBCF") # Recommendation Model generated by item-based collaboration filtering method
pred_ibcf <- predict(r_ibcf, getData(eval, "known"), type="ratingMatrix") # predictions based on IBCF | Predict returns predicted ratings on test data
as(pred_ibcf, "matrix")[1:10,1:10]# Check the predicted ratings of 10 users for first 10 jokes.

# Assessing prediction accurcay of the two models
error <- rbind( UBCF = calcPredictionAccuracy(pred_ubcf, getData(eval, "unknown")), # "unkown" are the held out ratings in the test set
                IBCF = calcPredictionAccuracy(pred_ibcf, getData(eval, "unknown")))
error
# unknown is the jokes not used in making predictions for test data
```

We see that User-based collaboration filtering achieves higher prediction accuracy than the item based prediction accuracy

```
> error
         RMSE      MSE      MAE
UBCF 4.676325 21.86801 3.70405
IBCF 5.334554 28.45746 4.19891
```

Summary

Model performance

- Classification error

Frequent Item-set

Recommendation Systems

Course Wrap Up

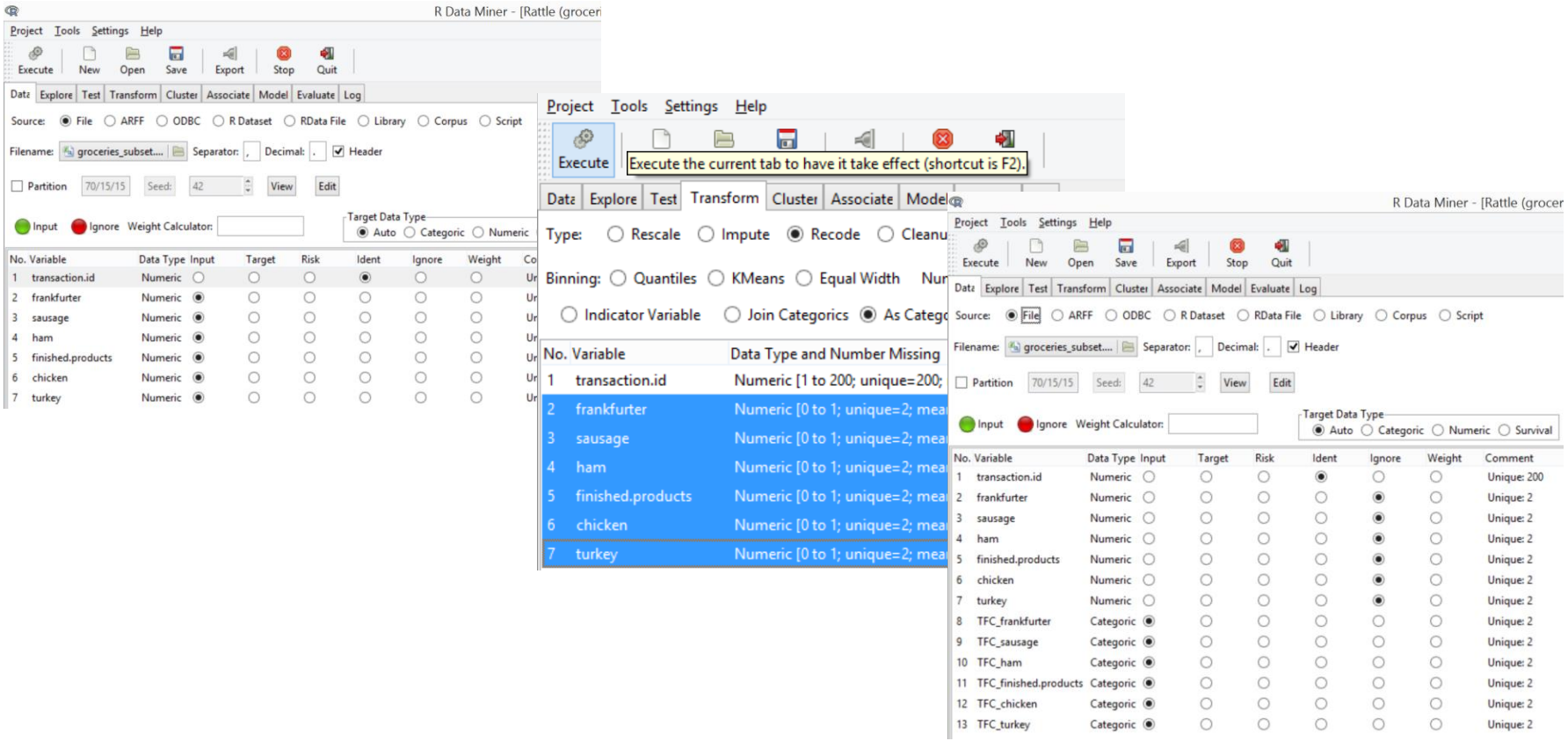
Groceries Data

Groceries dataset from Rattle

We use only a subset of the data for illustrative purposes. Data - groceries_subset.csv

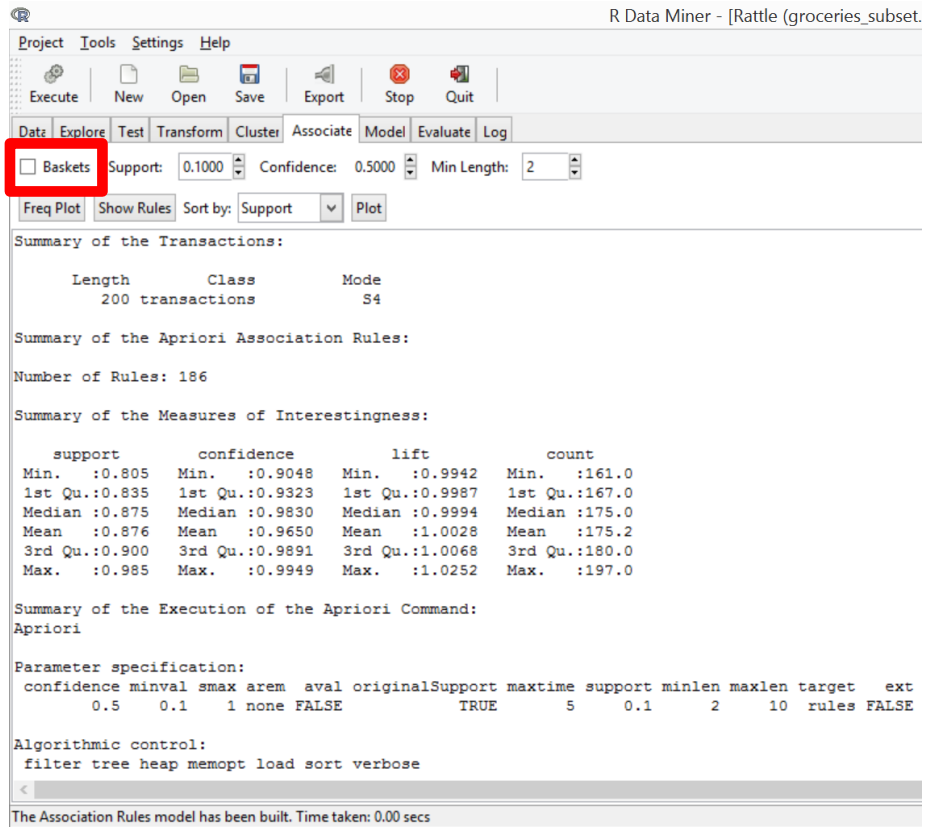
transaction id	frankfurter	sausage	ham	finished products	chicken	turkey
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0
12	0	0	0	0	0	0
13	0	0	0	0	0	0
14	1	0	0	0	0	0
15	0	0	0	0	1	0

Groceries_subset



Source: Rattle GUI / Togaware

Association Rule Mining



Source: Rattle GUI / Togaware

Frequency Plots

R Data Miner - [Rattle (groceries_subset.

Project Tools Settings Help

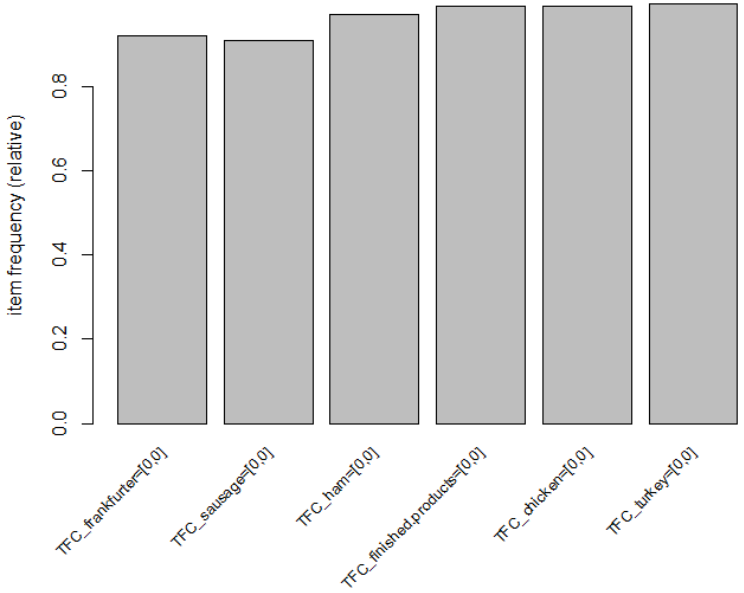
Execute New Open Save Export Stop Quit

Data Explore Test Transform Cluster Associate Model Evaluate Log

Baskets Support: 0.1000 Confidence: 0.5000 Min Length: 2

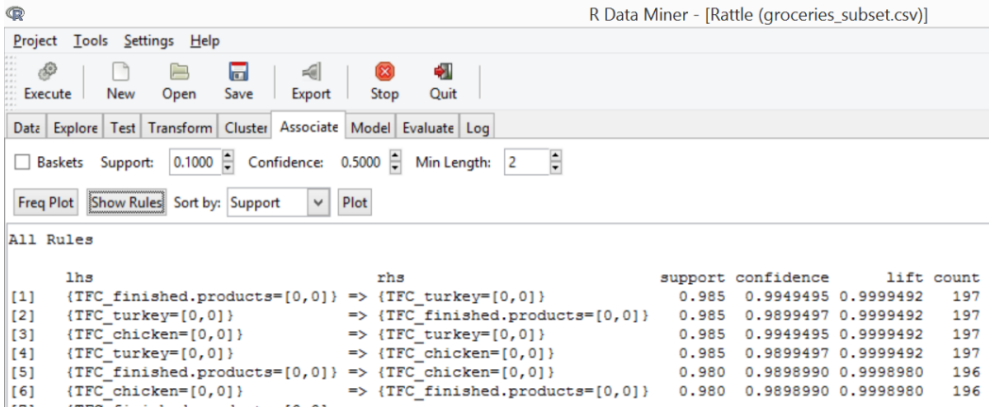
Freq Plot Show Rules Sort by: Support Plot

```
      Length        Class         Mode
         200 transactions           S4

Summary of the Apriori Association Rules:

Number of Rules: 186

Summary of the Measures of Interestingness:

    support        confidence          lift            count
 Min.   :0.805   Min.   :0.9048   Min.   :0.9942   Min.   :161.0
 1st Qu.:0.835   1st Qu.:0.9323   1st Qu.:0.9987   1st Qu.:167.0
 Median :0.875   Median :0.9830   Median :0.9994   Median :175.0
 Mean   :0.876   Mean   :0.9650   Mean   :1.0028   Mean   :175.2
 3rd Qu.:0.900   3rd Qu.:0.9891   3rd Qu.:1.0068   3rd Qu.:180.0
 Max.   :0.985   Max.   :0.9949   Max.   :1.0252   Max.   :197.0

Summary of the Execution of the Apriori Command:
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE       5     0.1      2     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE
```

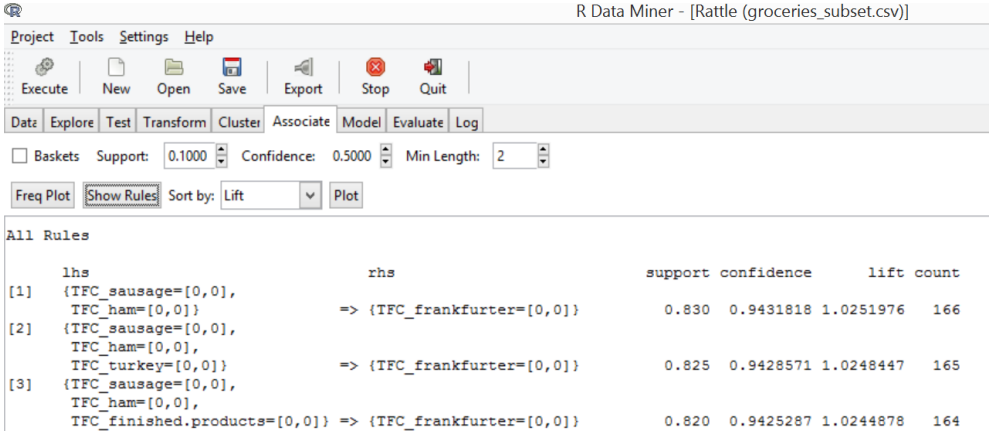


Source: Rattle GUI / Togaware

Association Rules Sorted on Support



Sorted on support



Sorted on lift

Source: Rattle GUI / Togaware

Plot – Display Rules Graphically

R Data Miner - [Rattle (groceries

Project Tools Settings Help

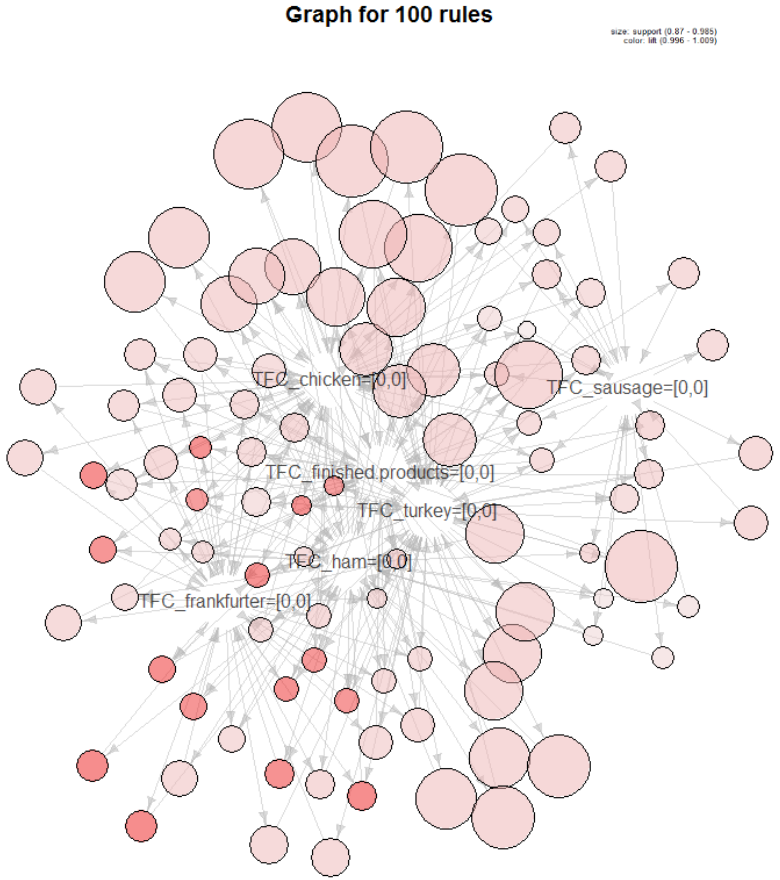
Execute New Open Save Export Stop Quit

Data Explore Test Transform Cluster Associate Model Evaluate Log

Baskets Support: 0.1000 Confidence: 0.5000 Min Length: 2

Freq Plot Show Rules Sort by: Support Plot

```
Summary of the Transactions:

      Length         Class          Mode
         200 transactions           S4

Summary of the Apriori Association Rules:

Number of Rules: 186

Summary of the Measures of Interestingness:

    support         confidence          lift            count
 Min.   :0.805   Min.   :0.9048   Min.   :0.9942   Min.   :161.0
 1st Qu.:0.835   1st Qu.:0.9323   1st Qu.:0.9987   1st Qu.:167.0
 Median :0.875   Median :0.9830   Median :0.9994   Median :175.0
 Mean   :0.876   Mean   :0.9650   Mean   :1.0028   Mean   :175.2
 3rd Qu.:0.900   3rd Qu.:0.9891   3rd Qu.:1.0068   3rd Qu.:180.0
 Max.   :0.985   Max.   :0.9949   Max.   :1.0252   Max.   :197.0

Summary of the Execution of the Apriori Command:
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target
        0.5    0.1    1 none FALSE            TRUE       5     0.1      2     10  rules
   ext
 FALSE
```



Source: Rattle GUI / Togaware

References

Hahsler, M. (2015). *recommenderlab: A framework for developing and testing recommendation algorithms*. Retrieved May 22, 2019, from https://cran.r-project.org/web/packages/recommenderlab/vignettes/recommenderlab.pdf

Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval*, 4(2).

Rattle GUI / Togaware (https://rattle.togaware.com/)

Shailesh, K. (2019) Unsupervised machine learning. In B. Pochiraju and S. Seshadri (Eds.), *Essentials of Business Analytics: An Introduction to the Methodology and its Applications* (p. 502). Springer.