Angoss
Predict. Act. Perform.™

WHITEPAPER

Comparing Statistical Models in R with KnowledgeSTUDIO

November 2012

Table of Contents

Executive Summary

This document will provide a summary of the differences, advantages and disadvantages of performing statistics modelling in R (R Development Core Team, 2012) and KnowledgeSTUDIO (Angoss Software, 2012). The statistical models explored in this document are Decision Trees, Logistic Regression and Linear Regression Models.

R is an open-source software program with a programming language interface, whereas KnowledgeSTUDIO is a commercial application based on graphical user interface. R is used for statistical computing and graphics, and various functions can be executed by downloading specific packages from CRAN (Comprehensive R Archive Network). KnowledgeSTUDIO features a data mining and predictive analytics workbench with graphics visualization and integration with other analytical environments in one package.

This document will demonstrate that KnowledgeSTUDIO is the superior software for training and scoring Decision Trees because of its fine-tuned controls and ability to manually select splits in a tree. The packages in R for building Decision Trees, rpart() and tree(), can only perform binary splits of data, have limited controls and do not allow the user to manually select splits. On the other hand, the difference in building Logistic and Linear Regression models In R compared to KnowledgeSTUDIO is minor. Both programs display similar outputs, and KnowledgeSTUDIO offers a few more options for model building. However, KnowledgeSTUDIO is more desirable for portability, because it can both export and import PMML code, whereas R can only export it.

Model	Function	R	KnowledgeSTUDIO
Decision Trees	Splitting Index	Up to 2 options	4 options
	Splitting Rules	1 option	2 options
	Tree type	Binary	Unlimited
	Data Type/Distribution	4 types	Unlimited
	User control of Tree	N/A	Yes
	Import/Export PMML	N/Y	Y/Y
Logistic and Linear Regressions	Variable Selection	3 options	5 options
	Interactions	Can be specified in model	Need to be created in dataset
	Adjusting Significance	N	Y

Preamble

This document is a guide aimed to help users navigate between R (R Development Core Team, 2012) and KnowledgeSTUDIO (Angoss Software, 2012). R is an open-source software program with a programming language interface used for statistical computing and graphics. Various functions in R can be executed by downloading specific packages from CRAN (Comprehensive R Archive Network). KnowledgeSTUDIO is a software package that provides user interface data analysis, graphics visualization, data mining workbench and integration with other analytical environments in one package. This document will provide a comparative analysis of how R and KnowledgeSTUDIO execute Decision Trees, Logistic Regressions and Linear Regressions. Each statistical model will be run in each program and PMML code will be and exported and imported to explore how the programs read each other's outputs.

1. Decision Trees

1.1 Decision Trees in R with rpart

To build Decision Trees in R, download the package rpart() (Therneau & Atkinson, 2012) from CRAN. The rpart()package uses CART to create binary trees. We will use the Infertility data set found in R to illustrate how to build decision trees. For reference, blue text represents commands, and black text represents the output generated from the blue text code. Comments are denoted with '##'.

First, let's take a look at the data. There are 6 different variables, and a summary can be called up using the `summary()` function:

```
> summary(infert)
   education        age            parity          induced         case     spontaneous     stratum
 0-5yrs : 12   Min.   :21.00   Min.   :1.000   Min.   :0.0000   0:165   0:141        Min.   : 1.00
 6-11yrs:120   1st Qu.:28.00   1st Qu.:1.000   1st Qu.:0.0000   1: 83   1: 71        1st Qu.:21.00
 12+ yrs:116   Median :31.00   Median :2.000   Median :0.0000           2: 36        Median :42.00
               Mean   :31.50   Mean   :2.093   Mean   :0.5726                        Mean   :41.87
               3rd Qu.:35.25   3rd Qu.:3.000   3rd Qu.:1.0000                        3rd Qu.:62.25
               Max.   :44.00   Max.   :6.000   Max.   :2.0000                        Max.   :83.00
 pooled.stratum
 Min.   : 1.00
 1st Qu.:19.00
 Median :36.00
 Mean   :33.58
 3rd Qu.:48.25
 Max.   :63.00
```

Because we will build a binary tree based on `case`, we need to tell R that this variable is a factor. We will also tell it that `spontaneous` is a factor:

```
> ## Set case as a binary factor
> infert$case<-as.factor(infert$case)
> infert$spontaneous<-as.factor(infert$spontaneous)
```

Next, fit a model to that data, with `case` as the binary dependent variable:

```
> inf.model1<-rpart(case~ age+ education+ induced+ spontaneous + stratum +
pooled.stratum, data=infert, parms = list(split = "gini"), method="class")
```

Because we are testing a binary variable, specifiy in the model `method="class"`, which is used for categorical data. The other options for parameter `method` are `"anova"` for continuous data, `"poisson"` for Poisson or count data and `"exp"` for exponential or survival data.

Although we used the default parameters for fitting this model, they can be changed using `rpart.control()`:

```
rpart.control(minsplit = 20, minbucket = round(minsplit/3), cp = 0.01,
maxcompete = 4, maxsurrogate = 5, usesurrogate = 2, xval = 10,
surrogatestyle = 0, maxdepth = 30, ...)
```

Where (directly taken from `rpart` manual):

`Minsplit` is the minimum number of observations that need to be in a node before a split is executed

`Minbucket` is the minimum number of observations in a terminal node.

`Cp` complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted.

`Maxcompete` number of competitor splits displayed in the output.

`Maxsurrogate` number of surrogate splits retained in output.

`Usesurrogate` how to use surrogates in the splitting process. 0 means display only; an observation with a missing value for the primary split rule is not sent further down the tree. 1 means use surrogates, in order, to split subjects missing the primary variable; if all surrogates are missing the observation is not split. For value 2, if all surrogates are missing, then send the observation in the majority direction.

`xval` number of cross-validations.

`surrogatestyle` controls the selection of a best surrogate. If set to 0 (default) the program uses the total number of correct classification for a potential surrogate variable, if set to 1 it uses the percent correct, calculated over the non-missing values of the surrogate. The first option more severely penalizes covariates with a large number of missing values.

`maxdepth` Set the maximum depth of any node of the final tree, with the root node counted as depth 0. Values greater than 30 rpart will give nonsense results on 32-bit machines.

After fitting the model, tell R to draw a tree with the results:

```
> plot(inf.model1, branch =0.4)
```

Here, `branch` refers to the shape of the branches from parent to child node. Please refer to the `rpart` manual for further instructions on how to shape the tree graphical output.

The output is as follows:

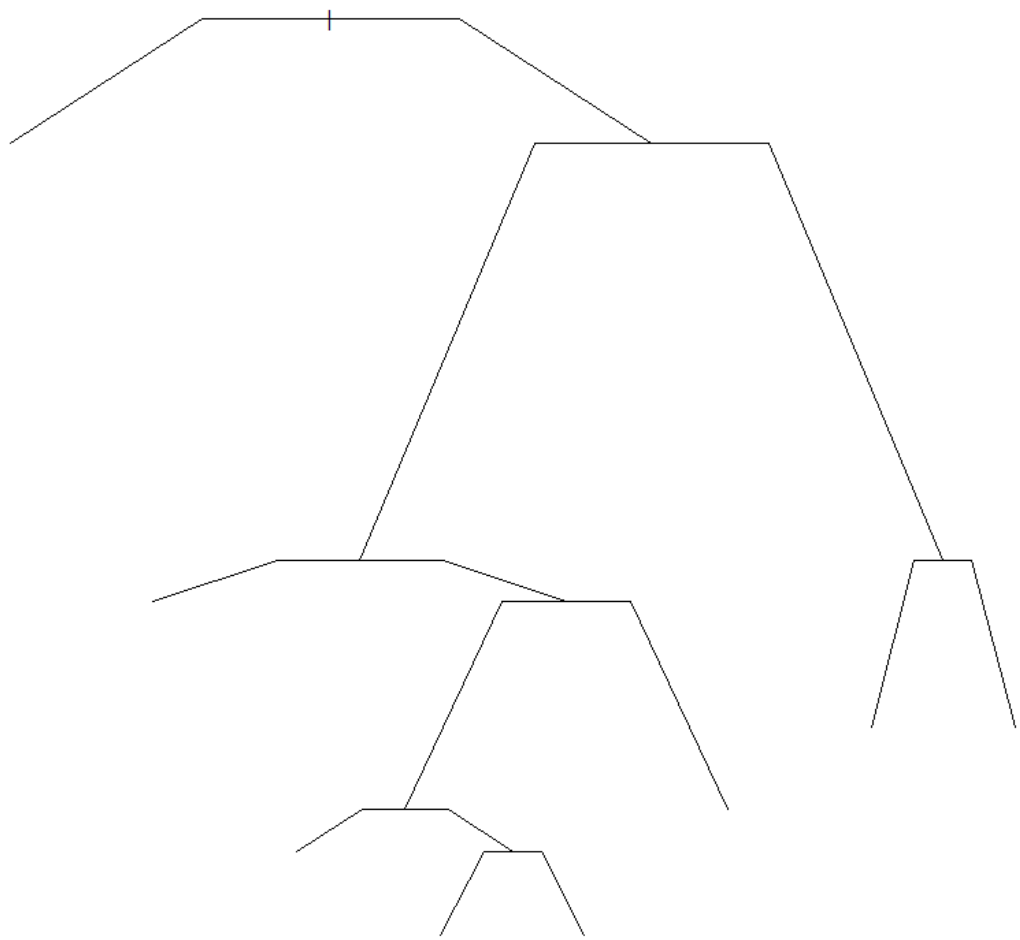


Figure 1: Bare decision tree generate from `inf.model1`.

To add labels to the tree:

```
> text(inf.model1, use.n=TRUE, all=T, fancy =T)
```

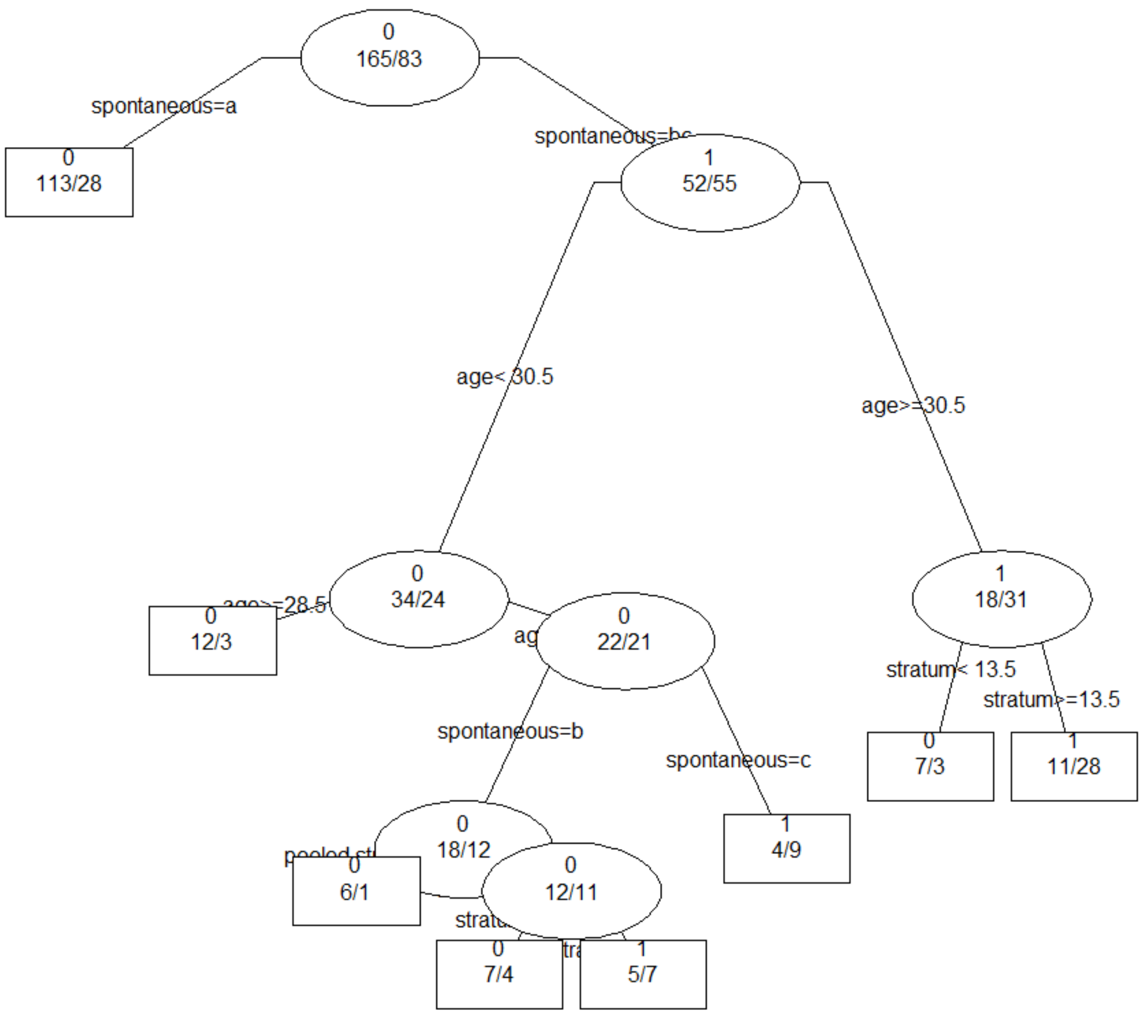


Figure 2: Final graphical output of the decision tree from `inf.model1`.

Since the tree is somewhat confusing to interpret, the summary of the splits can be viewed using the `summary.rpart()` function:

```
> ##Summary of model
> summary(inf.model1)
Call:
rpart(formula = case ~ age + education + induced + spontaneous +
    stratum + pooled.stratum, data = infert, method = "class")
  n= 248

          CP nsplit rel error    xerror       xstd
1 0.07831325      0 1.0000000 1.0000000 0.08953175
2 0.04819277      2 0.8433735 1.1566265 0.09241734
3 0.03012048      3 0.7951807 0.9638554 0.08869430
4 0.01204819      5 0.7349398 0.9277108 0.08778888
5 0.01000000      7 0.7108434 0.9879518 0.08926003

Node number 1: 248 observations,    complexity param=0.07831325
  predicted class=0  expected loss=0.3346774
    class counts:   165    83
   probabilities: 0.665 0.335
  left son=2 (141 obs) right son=3 (107 obs)
  Primary splits:
      spontaneous    splits as  LRR,       improve=12.106170000, (0 missing)
      pooled.stratum < 60.5 to the left,  improve= 0.026881720, (0 missing)
      induced        < 0.5  to the left,  improve= 0.024367570, (0 missing)
      stratum        < 73.5 to the left,  improve= 0.006766778, (0 missing)
      age            < 37.5 to the left,  improve= 0.005453149, (0 missing)
  Surrogate splits:
      induced        < 0.5  to the right, agree=0.605, adj=0.084, (0 split)
      stratum        < 72.5 to the left,  agree=0.601, adj=0.075, (0 split)
      pooled.stratum < 27.5 to the left,  agree=0.601, adj=0.075, (0 split)
      age            < 24.5 to the right, agree=0.581, adj=0.028, (0 split)
```
…etc.

Although only part of the output is copied here, this command displays all the splits that have been generated by the model.

Finally, to test this model in both R and KnowledgeSTUDIO, first train the model with 70% of the data:

```
> ##Train tree with 70% of data
> infer70<-read.table("Infertility 70.csv", sep=",", header=T)
> inf.model70<-rpart(case~ age+ education+ induced+ spontaneous + stratum +
pooled.stratum, data=infer70, method="class")
```

This package does not allow testing one model on a new data set that is of a different length. You can use the `predict()` function to predict the outcomes of the entire model, which can be compared later to the outcomes of the entire model in KnowledgeSTUDIO:

```
> predict(inf.model1)
            0         1
1   0.3076923 0.6923077
2   0.8014184 0.1985816
3   0.8014184 0.1985816
```
…etc.

Next, export this model in PMML code, using the `pmml` package (Graham et al., 2012):

```
> ##PMML code
> library(pmml)
>
> Inf.pmml<-pmml(inf.model70, model.name="Infertility Model 70%",
app.name="R",
+                 description="Infertility Decision Tree", dataset=infer70)
> # Describe the model in PMML and save it in an AML file
> xmlInf <- file.path("C:/Users/sstanescu/Documents/PMML codes","Inf Decision
Tree.xml")
> saveXML(Inf.pmml,xmlInf)

[1] "C:/Users/sstanescu/Documents/PMML codes/Inf Decision Tree.xml"
```

1.2 Decision Trees in R with tree

Decision trees in R can also be built using the package `tree` (Ripley, 2012). This package can also be downloaded from CRAN. First, download then load the package:

```
> ##Load tree package
> library(tree)
```

Then a build a model similar to the one built using `rpart`. Use the same Infertility data set, and set the variables `case` and `spontaneous` as factors:

```
> ## Set case as a binary factor
> infert$case<-as.factor(infert$case)
> infert$spontaneous<-as.factor(infert$spontaneous)
>
> ## Model tree
> inf.tree1<-tree(case~ age+ education+ induced+ spontaneous + stratum +
pooled.stratum, data=infert,
+                    method="recursive.partition")
```

In this tree, partition the data for `case`, based on `age`, `education`, `induced`, `spontaneous`, `stratum` and `pooled.stratum`. Notice that in this package compared to `rpart` the `method` has changed to "`recursive.partition`".

```
> ## Model tree
> inf.tree1<-tree(case~ age+ education+ induced+ spontaneous + stratum +
+   pooled.stratum, control=tree.control(nobs=248, mincut = 5, minsize = 10,
+   mindev = 0.01), data=infert, method="recursive.partition")
```

Next, plot this tree using the `plot()` function and add labels using the `text()` function:

```
> ##Plot tree
> plot(inf.tree1,  type = c("proportional", "uniform"))
> text(inf.tree1, splits=TRUE, label="yval")
```

The `control` function within the expression provides adjustments for tree building parameters:

`Nobs` is used to specify the number of observations used to create the tree. This allows for using only a partition of the data to build the tree.

`Mincut` is the minimum number of observations to use at each child node.

`Minsize` is the smallest node size (a weighted quantity).
`Mindev` the within node deviance must be at least this times that of the root node for the node to be split.

The above code produces the following output:

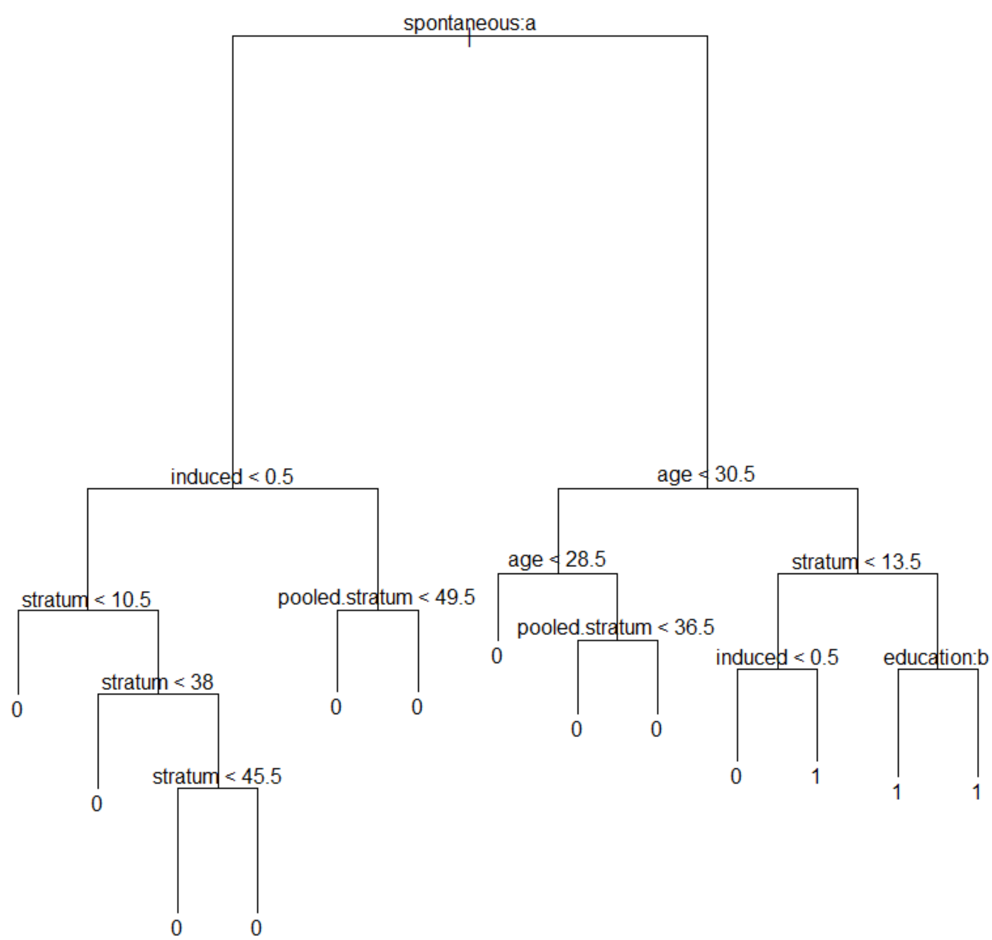


Figure 4: Tree output using the `tree` package in R.

Now, train the tree with 70% of the data:

```
> ##Train tree with 70% of data
> infer70<-read.table("Infertility 70.csv", sep=",", header=T)
> inf.tree70<-tree(case~ age+ education+ induced+ spontaneous + stratum +
pooled.stratum, data=infer70,
+                method="recursive.partition")
> ##Plot 70% tree
> plot(inf.tree70,  type = c("proportional", "uniform"))
> text(inf.tree70, splits=TRUE, label="yval")
```

The tree looks like:

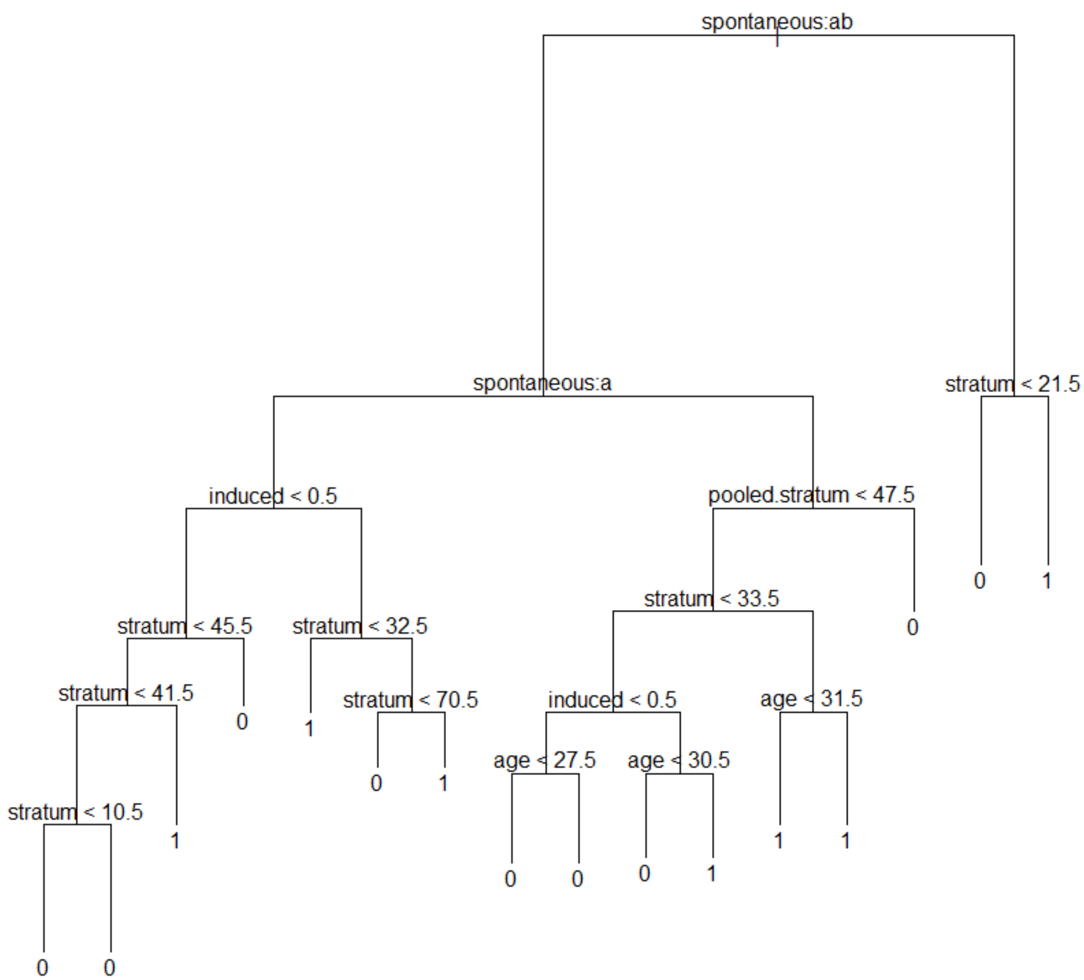


Figure 5: Tree created using only 70% of the data set.

The summary() command can be used to call up model details:

```
> summary(inf.tree1)

Classification tree:
tree(formula = case ~ age + education + induced + spontaneous +
    stratum + pooled.stratum, data = infert, control = tree.control(nobs =
248,
    mincut = 5, minsize = 10, mindev = 0.01), method = "recursive.partition")
Number of terminal nodes:  13
Residual mean deviance:  0.9656 = 226.9 / 235
Misclassification error rate: 0.2621 = 65 / 248
```

Now score the remaining 30% of the data using the predict() function:

```
> ##Use predict with 30% of data
> infer30<-read.table("Infertility 30.csv", sep=",", header=T)
> predict(inf.tree70, newdata=infer30)
      0         1
1  0.5000000 0.5000000
2  0.7142857 0.2857143
3  0.5000000 0.5000000
...etc.
```

We can also predict the case by adding a type="class" clause:

```
> predict(inf.tree70, newdata=infer30, type="class")
     [,1]
 [1,] "1"
 [2,] "0"
 [3,] "1"
```

…etc.

Unlike rpart, the predict() function in tree can return predicted values for a new data set with a different length. In addition, unlike rpart objects, tree objects cannot be converted into PMML code.

1.3 Decision Trees in KnowledgeSTUDIO

Using the same data set on Infertility found in R, construct a Decision Tree in KnowledgeSTUDIO.

After loading the data, change the Tree Training parameters to match those used in R.

In Options-> Tree Training and select -> Non-p value Gini Variance.

In Options-> Tree Growth set Minimum Node Creation Size and Auto-Grow Stop size to 1. These correspond to `minbucket` and `minsplit` respectively in R.

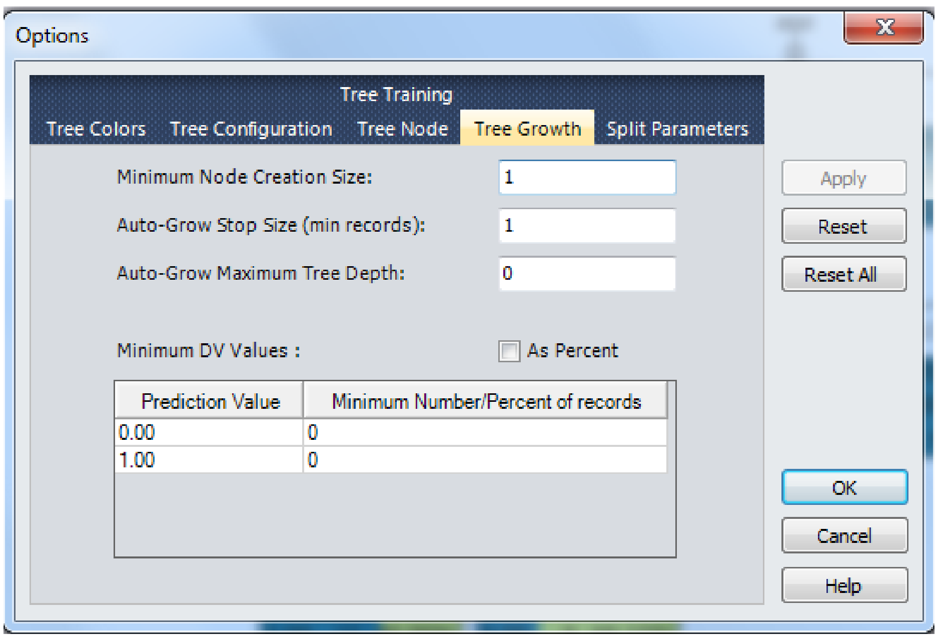


Figure 6: Tree Growth Dialog window in KnowledgeSTUDIO.

Since R only processes binary splits and KnowledgeSTUDIO can split into more than two categories, use Force Split by right-clicking on the first node, and indicate which variable each node should be split on. The Decision Tree in KnowledgeSTUDIO similar to the one created with `rpart` will look like:

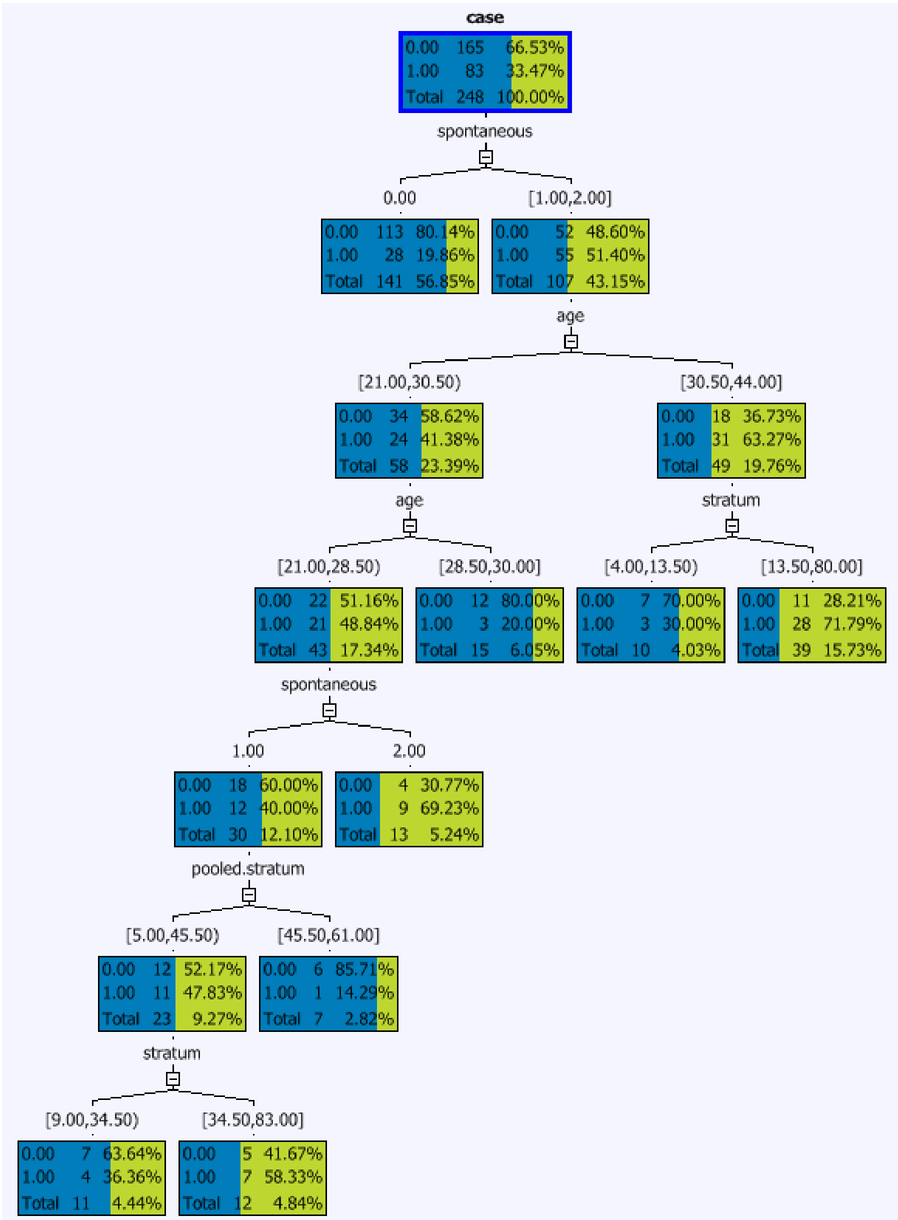


Figure 7: Decision Tree in KnowledgeSTUDIO.

Similarly, you can also build a tree to resemble the tree created using the `tree` package in R:

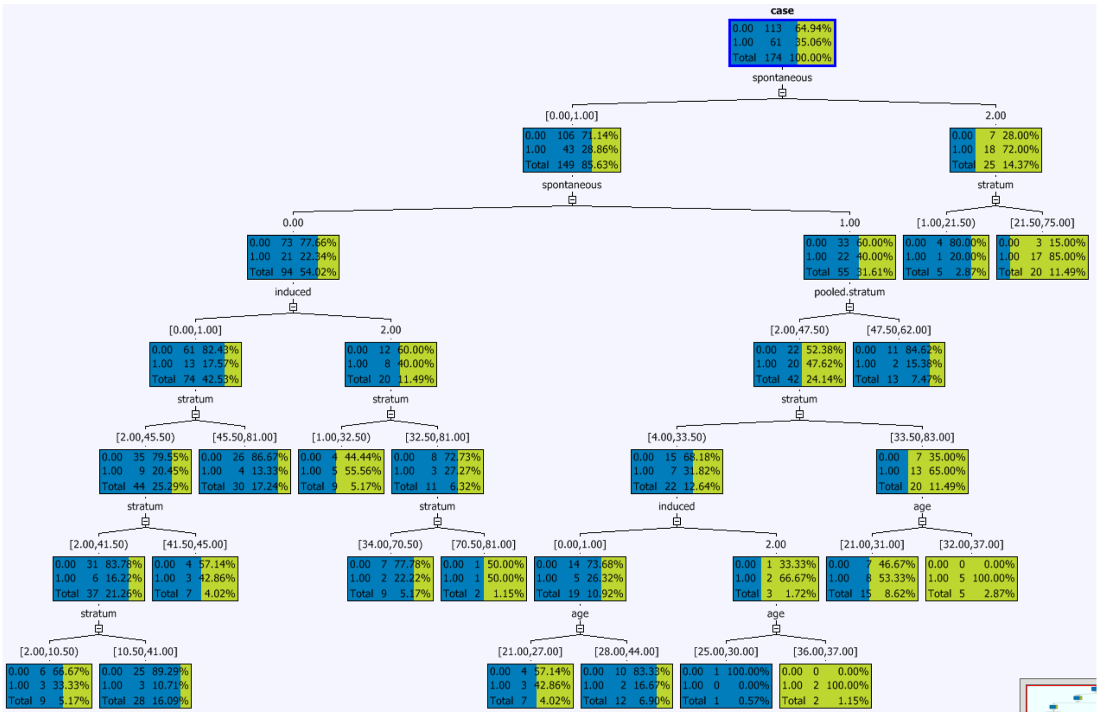
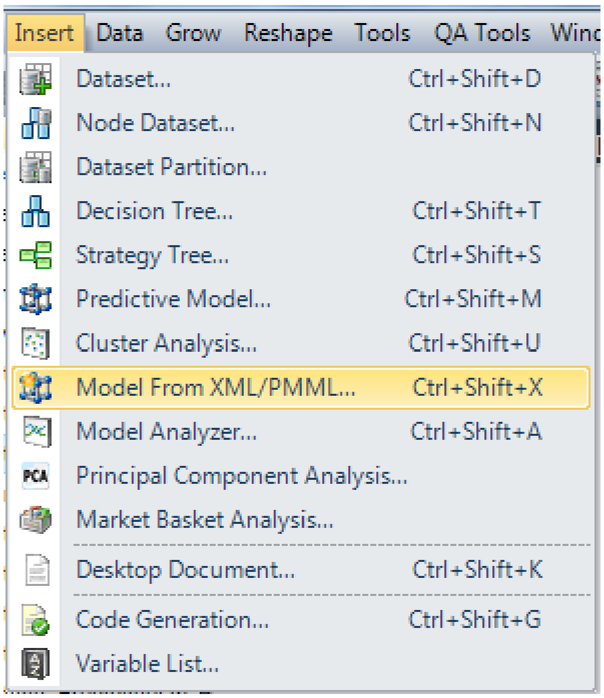


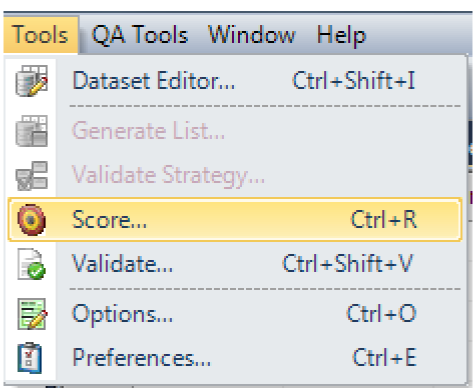
Figure 8: Decision tree in KnowledgeSTUDIO resembling tree made by `tree` package in R.

For the above tree (Fig. 8) trained on the 70% dataset partition, we score the 30% partition and compared it with the predicted values generated by R. About 27% of the predicted values (20/74) between `tree` in R and KnowledgeSTUDIO match.

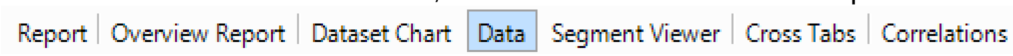
To compare the decisions trees generated by `rpart` in R and KnowledgeSTUDIO, first import the PMML for the decision tree created in R and score it. To do so, selected the model from XML/PMML and import the saved PMML file:



Then select Score from Tools to score the PMML model:



Once the model has been scored, click on the data tab to access predicted values:



These predicted values can now be compared to those generated by R. For this particular example, these values match. Furthermore, you can also score the KnowledgeSTUDIO decision tree using the same method as above, and obtain the same predicted values.

1.4 Tree Summary

	Control Functions		
Definition	**`Rpart`**	**`Tree`**	**KnowledgeSTUDIO**
Minimum number of observations at each child node	`minsplit`	`mincut`	Auto-Grow Stop Size
Smallest node size	`minbucket`	`minsize`	Min. Node Creation Size
Number of competitor splits displayed in the output	`maxcompete`	-	-
Within node deviance must be at least this x the root node for the node to be split.	-	`mindev`	-
Number of levels to grow the tree	`maxdepth`	-	Auto-Grow Maximum Depth
Complexity parameter	`cp`	-	Critical value, error complexity, reduced error
Export/Import PMML?	`Y/N`	`N/N`	Y/Y

Split Functions and Parameters			
Split Search Method	-	-	Cluster (maximize similarity)
Split Search Method	-	-	Exhaustive (maximize statistical significance)
Splitting Index	-	-	Unadjusted Raw p-value
Splitting Index	-	-	Bonferroni Adjusted p-value
Splitting Index	`information`	-	Entropy variance/information
Splitting Index	`Gini variance`	`Gini variance`	Gini variance
Splitting Index	-	`Deviance`	-

Type of data type/ distributions	`Method =`
	
	`“class”–categorical`
	`“anova”-continuous`
	`“poisson”-count`
	`“exp”-exponential`
		`Type = “recursive. partioning”`	Any – no assumptions

Predictions	-	`predict()`	Scoring models

1.5 Comparison Summary

KnowledgeSTUDIO gives the user more fine-tuned options in Decision Tree training compared to R. In R, Decision Trees can be built using the rpart() and tree() packages found in CRAN. The ‘goodness of fit’ of a Decision Tree model is based on its ability to predict new values based on the model trained on previous data. Thus, Decisions Trees go through a two-step process: they must first be trained on old data, and then used to predict new data. Overall, rpart() has finer controls for training Decision Trees compared to tree(), however KnowledgeSTUDIO has the most options for Decision Tree training. For example, KnowledgeSTUDIO allows the user to specify whether the splitting search method in the Decision Tree will create groups that maximizes similarity within groups or maximizes statistical significance, whereas the only option available for rpart() and tree() maximizes statistical significance. In addition, KnowledgeSTUDIO has four options for splitting indexes, whereas rpart() and tree() have one and two splitting index options respectively. Moreover, both rpart() and tree() can only execute binary splits in Decision Trees, whereas KnowledgeSTUDIO is not limited in the number of splits it can perform. In KnowledgeSTUDIO, the user can also manually select splits in a tree for a specific variable, while this is not possible in rpart() or tree(). In addition, the tree output generated by rpart() and tree() are visually harder to interpret, and the user has to create additional code to display labels on the tree.

KnowledgeSTUDIO is capable of predicting new values from old data in models created both in KnowledgeSTUDIO and those imported from PMML. For predicting new data, rpart() cannot predict values for a new dataset that is smaller than the dataset that the Decision Tree was trained on, whereas this is not an issue in tree() or KnowledgeSTUDIO. Finally, KnowledgeSTUDIO can both export and import PMML code. rpart() can only export and not import, and tree() cannot export or import PMML code. Thus, KnowledegSTUDIO is a superior choice for implementing Decision Trees because of its fine-tuned controls, predictive ability, and portability through PMML.

2 Logistic Regression Models

2.1 Logistic Regression Models in R

To explore logistic models in R, we will use the same data set on Infertility that was used for creating the Decision Trees.

First, fit the model to the Infertility data using the `glm()`function in R, with `case` as the dependent variable, then ask for the `summary()`of the model. Note that the function `glm()`does not necessarily assume that you are fitting a logistic model. You must specify `family=binomial` (Note: the dependent variable is on the left of the '~', and the independent variables on the right. The '+' signs indicate the variables that are considered independently, ':' represents an interaction, and '*' represents a cross between variables):

```
> infer.glm<-glm(case~ education+ age+ parity+ induced+ spontaneous+ stratum
+            + pooled.stratum, family=binomial, data=infert)
>
> summary(infer.glm)

Call:
glm(formula = case ~ education + age + parity + induced + spontaneous +
    stratum + pooled.stratum, family = binomial, data = infert)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8100  -0.7883  -0.4585   0.8577   2.8985

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      -4.018966   2.143974  -1.875   0.0609 .
education6-11yrs  1.328579   1.567429   0.848   0.3967
education12+ yrs  3.505146   2.969647   1.180   0.2379
age               0.078161   0.038255   2.043   0.0410 *
parity           -0.452994   0.277247  -1.634   0.1023
induced           1.436517   0.320820   4.478 7.55e-06 ***
spontaneous1      2.162651   0.419756   5.152 2.58e-07 ***
spontaneous2      4.401187   0.680465   6.468 9.94e-11 ***
stratum          -0.002893   0.014627  -0.198   0.8432
pooled.stratum   -0.078942   0.043832  -1.801   0.0717 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 316.17  on 247  degrees of freedom
Residual deviance: 254.52  on 238  degrees of freedom
AIC: 274.52

Number of Fisher Scoring iterations: 4
```

Next, fit the same model with only a randomly selected subset of 70% of the data, which will be used to score the remainder 30% of the data in R and KnowledgeSTUDIO (Note: the 70/30 split of the data was performed in KnowledgeSTUDIO, which will be explained in the next section):

```
> ##With 70% of the data
> infer70<-read.table("Infertility 70.csv", sep=",", header=T)
> infer.glm1<-glm(case~ education+ age+ parity+ induced+ spontaneous+ stratum
+                  + pooled.stratum, family=binomial, data=infer70)
> summary(infer.glm1)

Call:
glm(formula = case ~ education + age + parity + induced + spontaneous +
    stratum + pooled.stratum, family = binomial, data = infer70)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5701  -0.8031  -0.4512   0.8521   2.5855

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      -4.582780   2.429566  -1.886   0.0593 .
education12+ yrs  4.367212   3.475084   1.257   0.2089
education6-11yrs  1.748251   1.779118   0.983   0.3258
age               0.078316   0.044637   1.754   0.0793 .
parity           -0.453801   0.325847  -1.393   0.1637
induced           1.659534   0.387939   4.278 1.89e-05 ***
spontaneous       2.399112   0.418035   5.739 9.52e-09 ***
stratum           0.004057   0.017585   0.231   0.8175
pooled.stratum   -0.095579   0.052761  -1.812   0.0701 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 225.43  on 173  degrees of freedom
Residual deviance: 177.29  on 165  degrees of freedom
AIC: 195.29

Number of Fisher Scoring iterations: 5
```

Now score the remained of the data using the model constructed on 70% of the data. The output shows log-odds for each data point, which are probabilities on a logit scale:

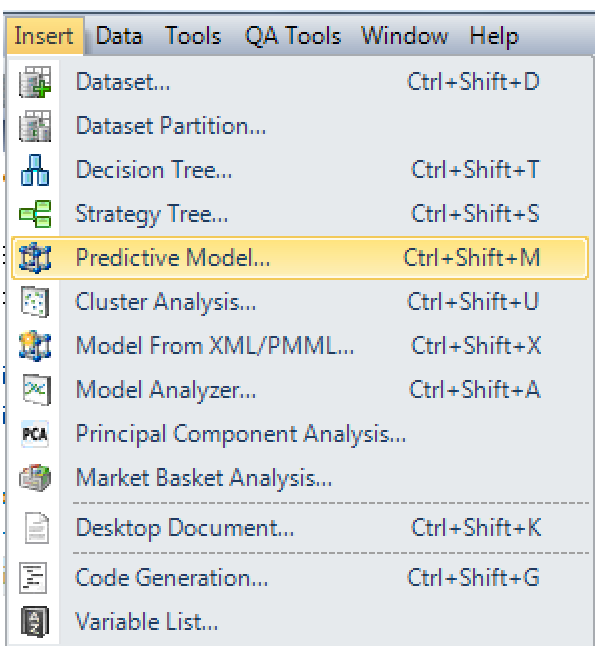
```
> ##Score the 70%  training model on the 30% subset of data
> ##Log-odds (probabilities on a logit scale)
> score1<-predict(infer.glm1, newdata=infer30)
> score1=as.matrix(score1)
> score1
          [,1]
1   -1.30234875
2   -3.32471321
3   -1.67838536
```
…etc.

Finally, export the PMML code for the training model (i.e. the model fitted to 70% of the data set):

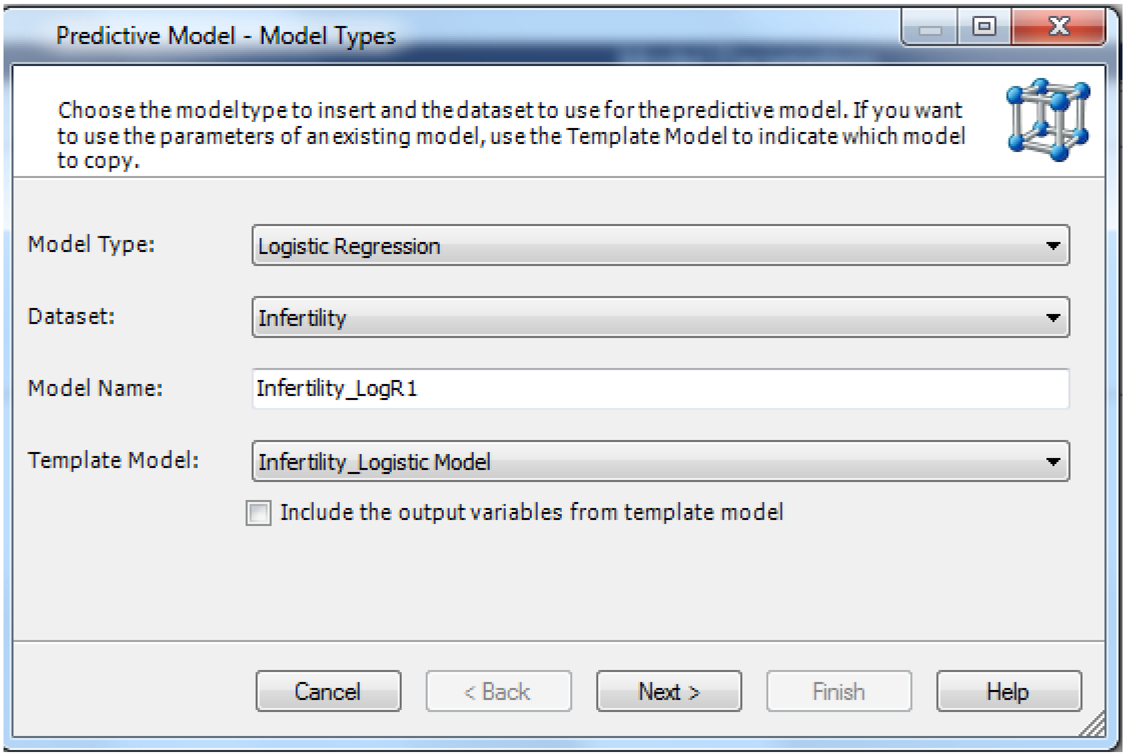
```
> ##70% logistic regression
> Infer.glm1.pmml<-pmml(infer.glm1, model.name="Infertility GLM 70%",
app.name="R",
+                        description="Infertility GLM 70%", dataset=infer70)
>
> # Describe the model in PMML and save it in an XML file
> Infer.glm1.xml <- file.path("C:/Users/sstanescu/Documents/PMML
codes","Infertility GLM 70%.xml")
> saveXML(Infer.glm1.pmml,Infer.glm1.xml)
[1] "C:/Users/sstanescu/Documents/PMML codes/Infertility GLM 70%.xml"
```

2.2 Logistic Regression Models in KnowledgeSTUDIO

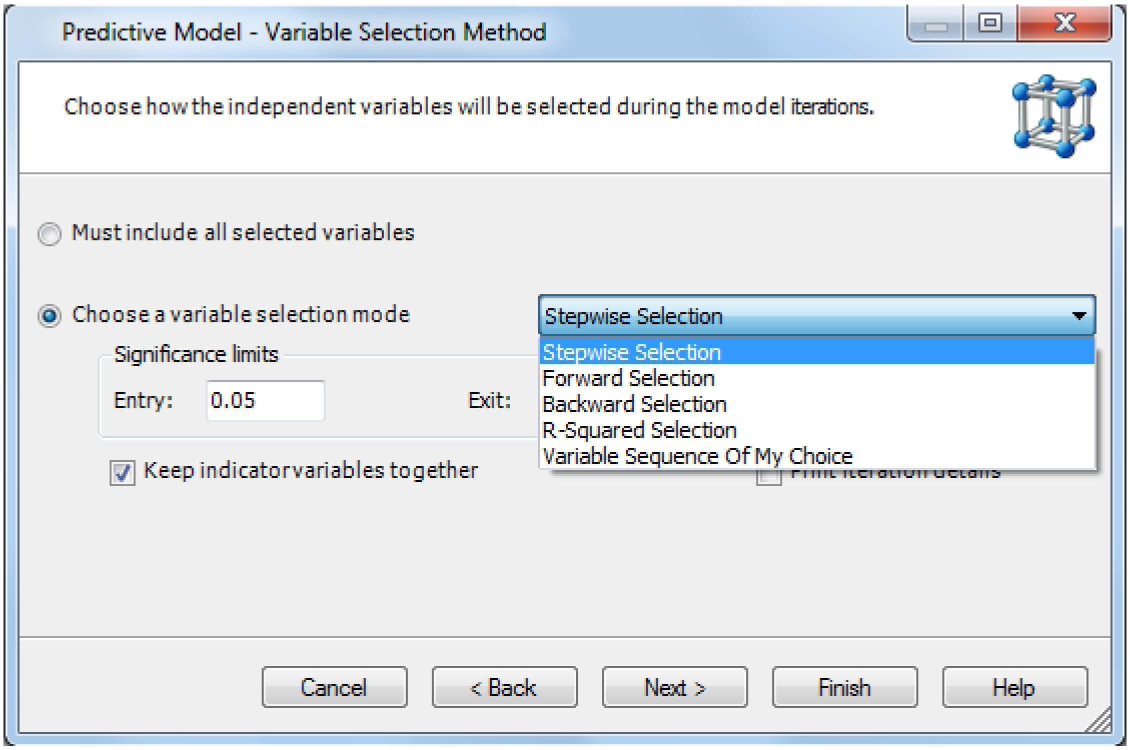
Using the same Infertility data set as before, insert a Logistic Regression in KnowledgeSTUDIO. Once the data has been uploaded, select “Predictive Model” from the “Insert” menu:



Then, select “Logistic Regression” from the drop-down menu from “Model Type”:



Although the default option is to include all variables in the model, you can select which way the variables are analyzed and which variables are used using the 'Variable Selection Method':



The following is the output of the model:

Output to view: Currently Selected Sequence Current Status: Model is trained. This is equivalent to sequence 1

Model Fitting Summary for case

Chi-Square: 61.640149
P-Value: 0.000000
Entropy Explained: 0.194958

Chi-Square Degrees Of Freedom: 8
AIC: 272.530962
BIC: 304.151821

	Negative 2(Log-Likelihood)	DF
Intercept Only	316.171111	-
Full Model	254.530962	8

Independent Variable Statistics

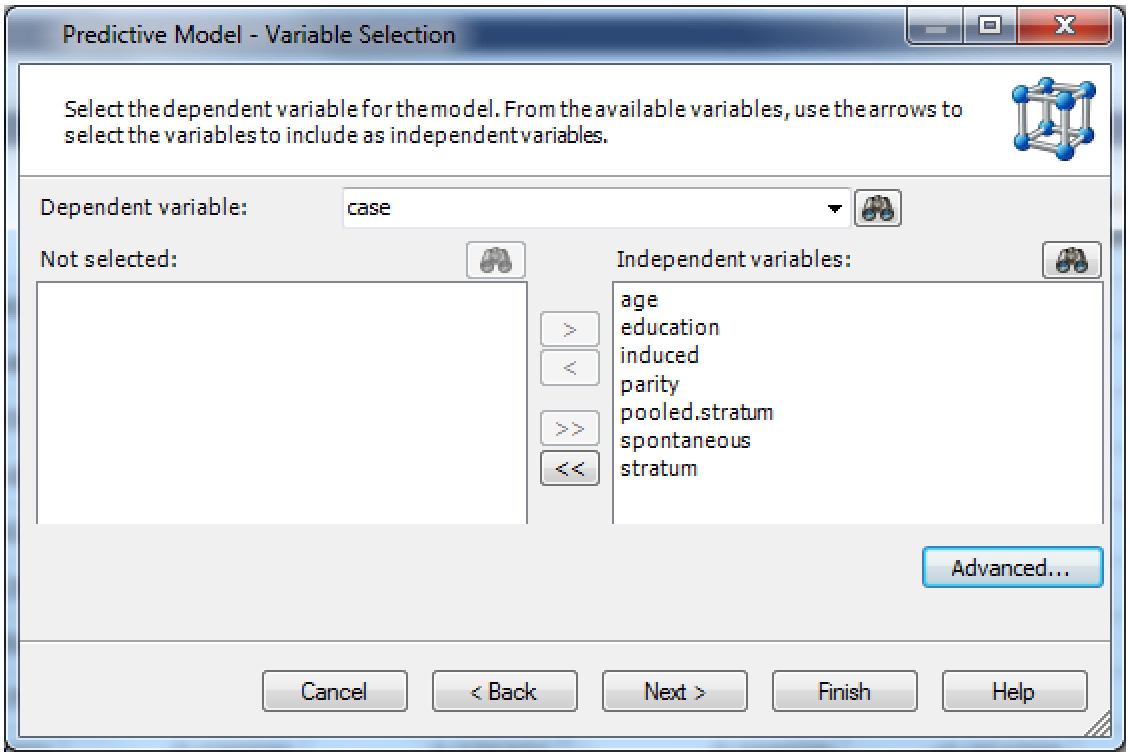
case = 1

Variable Name / Value		DF	Model Parameters	Parameter Standard Error	Wald Chi-Square	Significance	-95%	Odds Ratio	+95%	Standardized Parameter	Standardized Parameter Error
education	12+ yrs	1	3.489694	2.966030	1.384279	0.239374	0.098000	32.776000	>999.999000	1.744733	1.482918
	6-11yrs	1	1.320469	1.565708	0.711270	0.399023	0.174000	3.745000	80.581000	0.661225	0.784029
age		1	0.078590	0.038063	4.263186	0.038947	1.004000	1.082000	1.166000	0.412718	0.199888
parity		1	-0.451423	0.276888	2.658029	0.103028	0.370000	0.637000	1.096000	-0.564958	0.346526
induced		1	1.435628	0.320903	20.014065	0.000008	2.240000	4.202000	7.882000	1.060149	0.236973
spontaneous		1	2.191281	0.329107	44.332455	0.000000	4.694000	8.947000	17.053000	1.605204	0.241084
stratum		1	-0.002842	0.014622	0.037770	0.845905	0.969000	0.997000	1.026000	-0.068111	0.350462
pooled.stratum		1	-0.078768	0.043803	3.233650	0.072140	0.848000	0.924000	1.007000	-1.360492	0.756571
#INTERCEPT#		1	-4.039293	2.135930	3.576321	0.058609	<0.001000	0.018000	1.159000	-	-

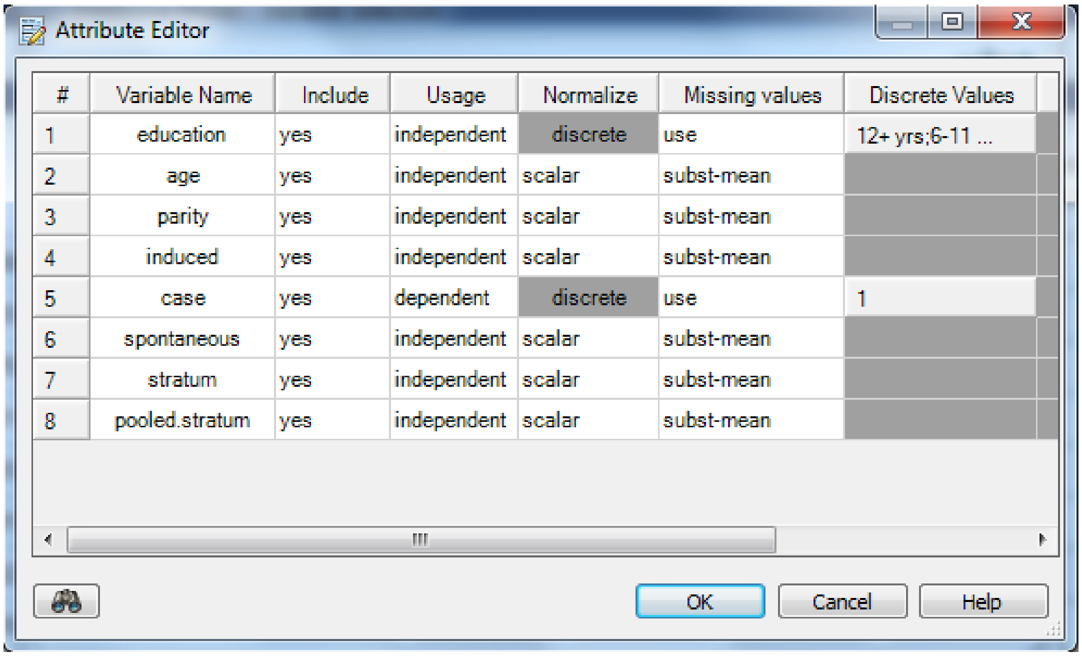
Variance Inflation Factors

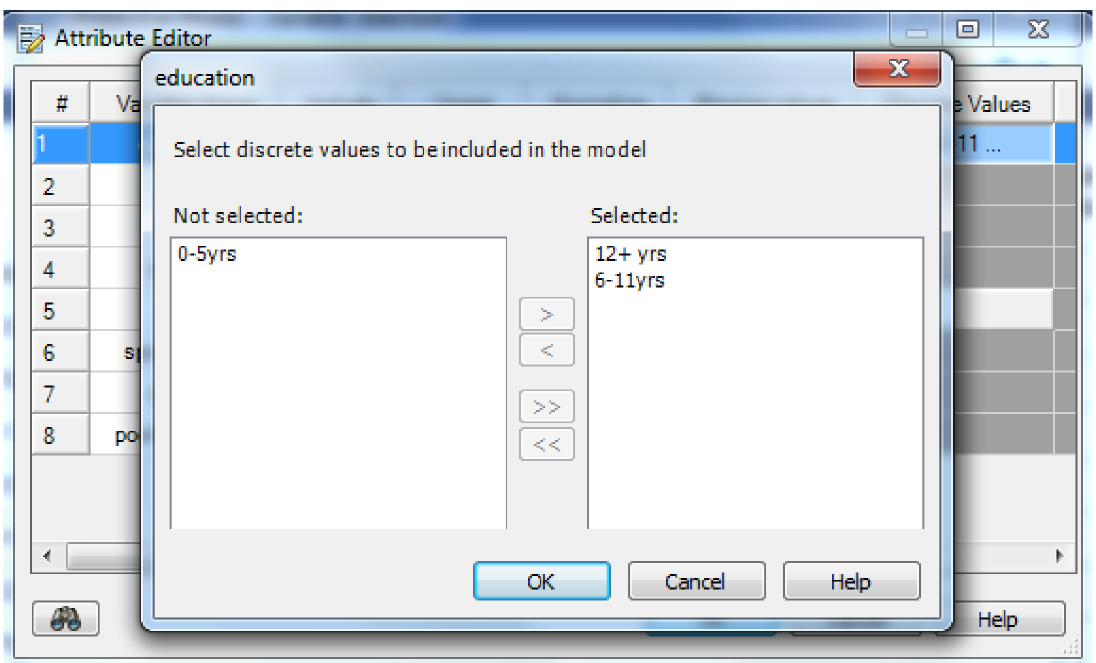
Variable	Value
([education]=='12+ yrs')	93.272
([education]=='6-11yrs')	27.900
[age]	1.713
[parity]	5.549
[induced]	1.846
[spontaneous]	1.643
[stratum]	5.151
[pooled.stratum]	22.966

Note – for the variable ‘Education’, KnowledgeSTUDIO initially used as reference the level ‘6-11 yrs’, whereas R used the level ‘0-5 yrs’ as reference. You can change what KnowledgeSTUDIO uses as reference in the “Advanced” Menu during model training:



Then select the variable you want to change:

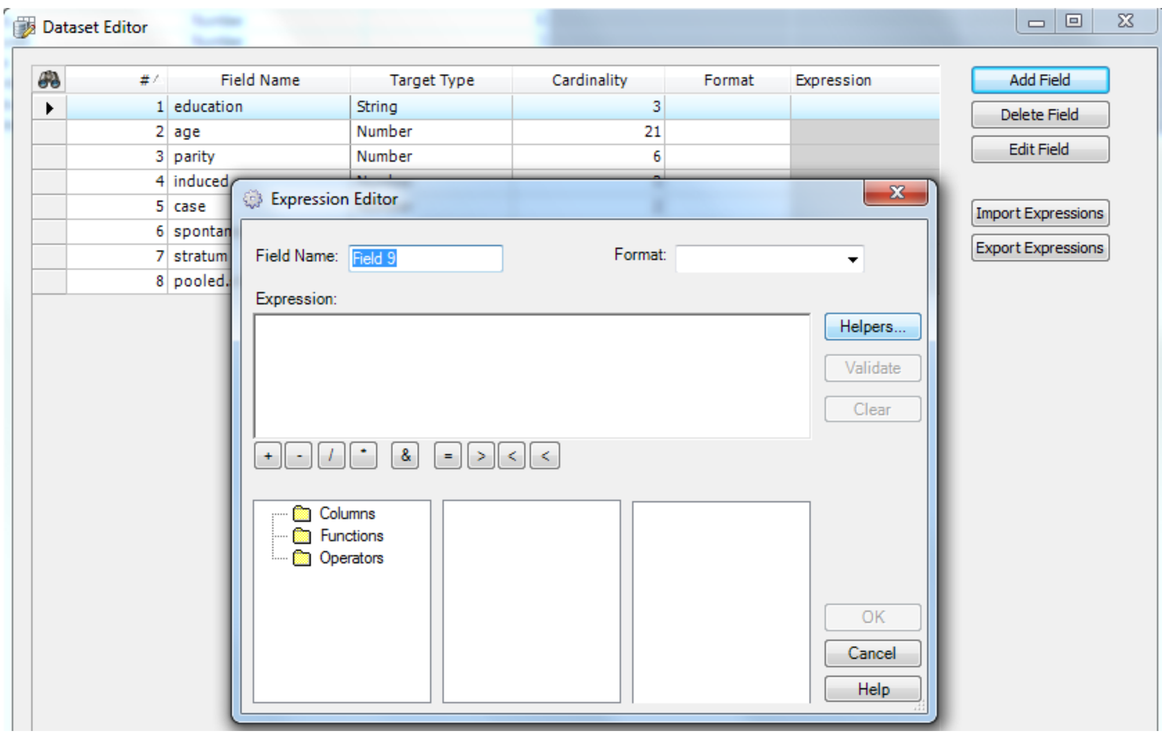




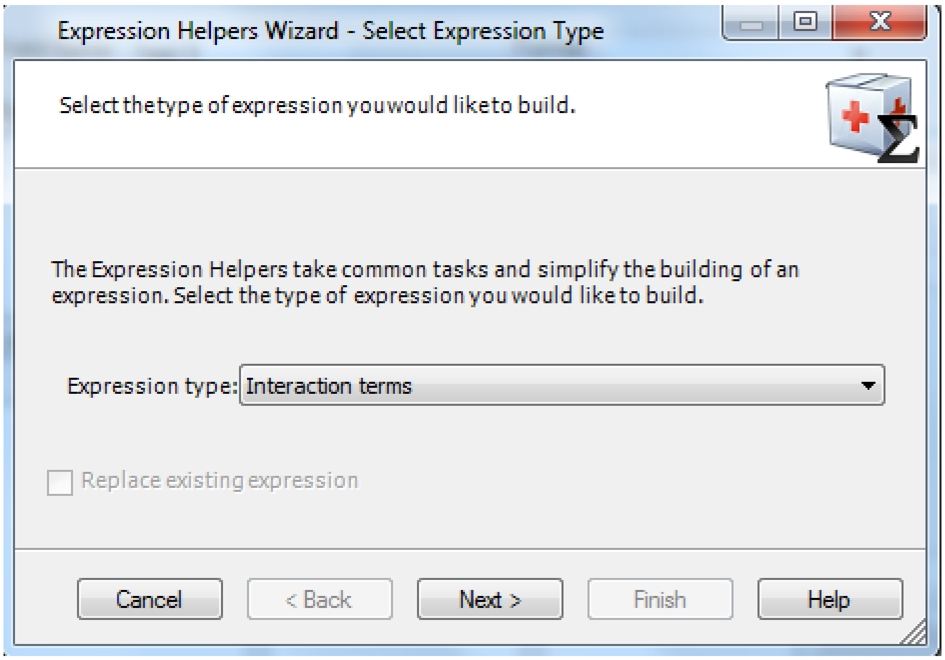
The value that will be used as a reference will go in the “Not Selected” column on the left, whereas the other values will go in the “Selected” column. In this particular model, ‘0-5’ years is not selected to match the model output created previously in R.

Once this option has been selected, now the models generated in R and KnowledgeSTUDIO are the same.

Note: To add interactions between variables, you must add new variables in the data using "Add Field" in the Dataset Editor:

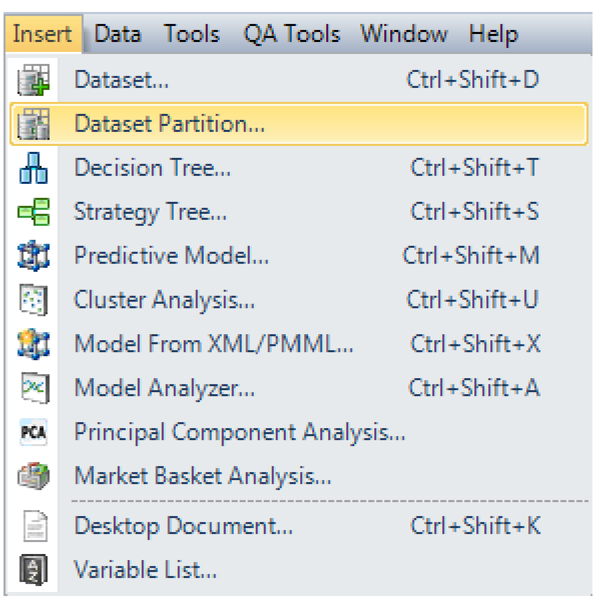


In this window, select “Helpers”, and set “Interaction terms” as the expression type:

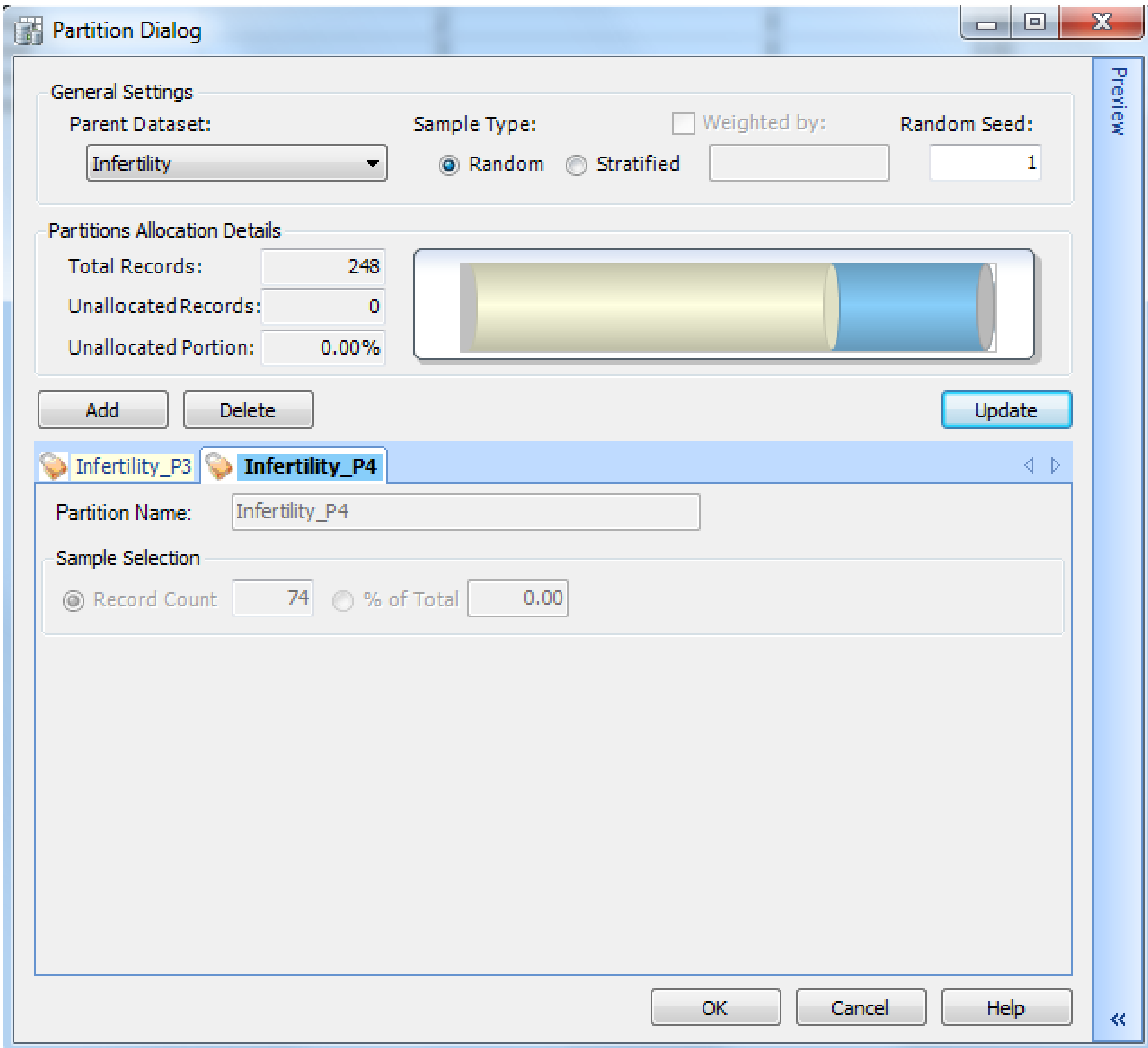


Now you can create interaction terms for your model.

To score this model, first partition the data in a random 70/30 split. To do, so open the Infertility data set and from the “Insert” menu, and select “Dataset Partition”:



Then indicate the partition as random, with the first partition set to 70%, and the second specified to contain the remaining 74 records (which was not exactly 30% - inputting 30% in the percent box will not get the desired result):



Next, import the 70% logistic model created earlier in R using “Insert” -> “Model from XML/PMML” and apply it to the 30% partitioned data. As before, to score the model, got to “Tools” -> “Score” to get predicted values. Now, you can compare the predicted values for the data scored in R and in KnowledgeSTUDIO. In this example, the difference between the predicted values is negligible.

3 Linear Regression Models

3.1 Linear Regression Models in R

To explore linear models in R, use the same Infertility data set as you used in the Decision Trees and Logistic Regressions.

Fit a linear model to the data using the `lm()`function, using `age` as the dependent variable, and use the `summary()` function to display the model output:

```
> ##Linear regression
> infer.lm<-lm(age~ education+ case+ parity+ induced+ spontaneous+ stratum
+              + pooled.stratum, data=infert)
> summary(infer.lm)

Call:
lm(formula = age ~ education + case + parity + induced + spontaneous +
    stratum + pooled.stratum, data = infert)

Residuals:
    Min      1Q  Median      3Q     Max
-9.3580 -2.6674 -0.9729  1.9094 11.5328

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       46.69067    2.06204  22.643  < 2e-16 ***
education6-11yrs -21.52899    2.33315  -9.227  < 2e-16 ***
education12+ yrs -44.29212    4.07077 -10.881  < 2e-16 ***
case1              1.11051    0.61391   1.809 0.071728 .
parity            -2.52339    0.46013  -5.484 1.06e-07 ***
induced           -1.73623    0.48245  -3.599 0.000389 ***
spontaneous1      -2.40339    0.66426  -3.618 0.000362 ***
spontaneous2      -3.19162    1.05932  -3.013 0.002867 **
stratum            0.06663    0.02399   2.777 0.005920 **
pooled.stratum     0.60192    0.06020   9.999  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.045 on 238 degrees of freedom
Multiple R-squared: 0.4284,    Adjusted R-squared: 0.4068
F-statistic: 19.82 on 9 and 238 DF,  p-value: < 2.2e-16
```

As before, fit the same model to 70% of randomly selected subset of the data, then score it on the remaining 30% using the `predict()` function (Note: `Type="response"` gives the predictions of `age`. The other option is `terms,` which provides an output of model terms):

```
> ##70% linear regression
> infer2.lm<-lm(age~ education+ case+ parity+ induced+ spontaneous+ stratum
+               + pooled.stratum, data=infer70)
> linear.prediction<-predict(infer2.lm, newdata=infer30, type="response")
> linear.prediction=as.matrix(linear.prediction)
> linear.prediction
       [,1]
1  30.44264
2  33.08635
3  31.98258
```

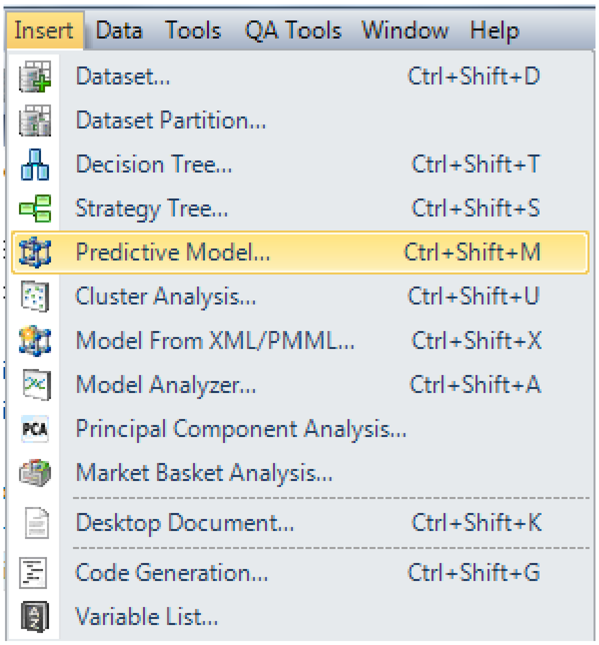
…etc.

Finally, export this model to PMML code as follows:

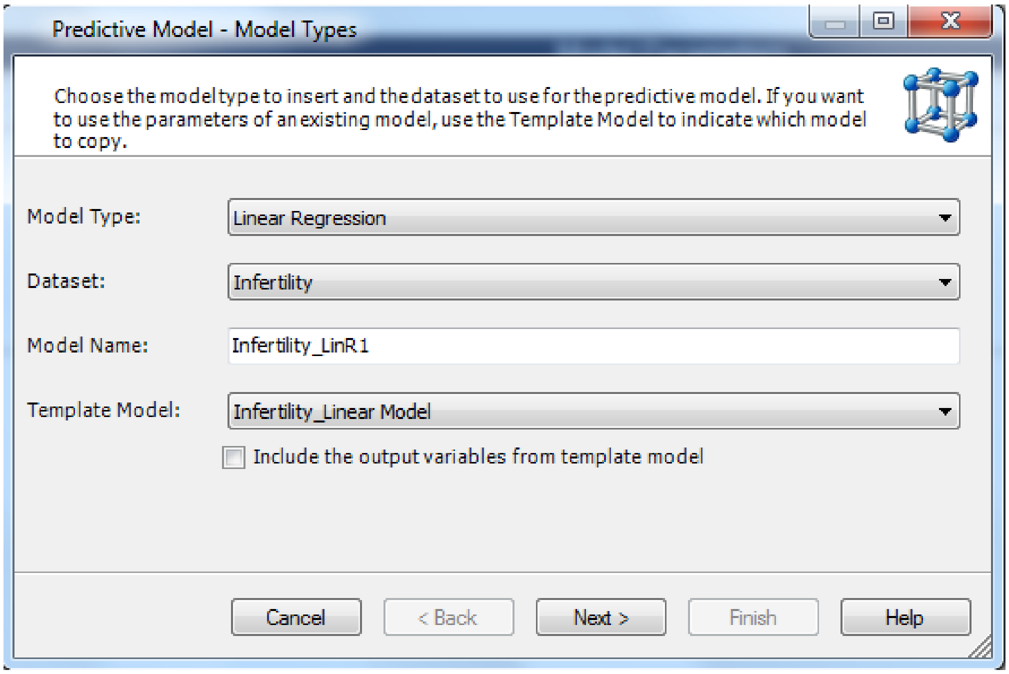
```
> ##Create PMML code for 70%
> Infer.lm2.pmml<-pmml(infer.lm, model.name="Infertility Linear Model 70%",
app.name="R",
+                      description="Infertility LM 70%", dataset=infert)
>
> # Describe the model in PMML and save it in an XML file
> Infer.lm2.xml <- file.path("C:/Users/sstanescu/Documents/PMML
codes","Infertility Linear Model 70%.xml")
> saveXML(Infer.lm2.pmml,Infer.lm2.xml)
[1] "C:/Users/sstanescu/Documents/PMML codes/Infertility Linear Model
70%.xml"
```

3.2 Linear Models in KnowledgeSTUDIO

Similar to inserting a Logistic Model, inserting a Linear Model can be executed through the "Insert" menu. From the options, select "Predictive Model":



Then select “Linear Regression” from the “Model Type” drop-down menu:



The model gives the following output:

Analysis of Variance for age

Variance Explained:	0.424334	**Adjusted Variance Explained:**	0.405065
F-Ratio:	22.021450	**F-Ratio Degrees Of Freedom 1 / 2:**	8/239
P-Value:	0.000000	**AIC:**	702.674266
		BIC:	734.295125

	Sum-Of-Squares	DF	Mean-Square
Regression	2,890.564336	8	361.320542
Error	3,921.431632	239	16.407664
Total	6,811.995968	247	27.578931

Independent Variable Statistics

age

Variable Name / Value		DF	Model Parameters	Parameter Standard Error	Wald Chi-Square	Significance	Standardized Parameter	Standardized Parameter Error
education	12+ yrs	1	-44.849678	4.053764	122.405748	0.000000	-4.269846	0.385933
	6-11yrs	1	-21.826289	2.325170	88.115140	0.000000	-2.081194	0.221711
parity		1	-2.507653	0.460623	29.637655	0.000000	-0.597601	0.109771
induced		1	-1.790878	0.481290	13.845816	0.000198	-0.251827	0.067677
case	1	1	1.138480	0.614409	3.433489	0.063887	0.102505	0.055319
spontaneous		1	-1.832311	0.497979	13.538693	0.000234	-0.255589	0.069463
stratum		1	0.067843	0.024007	7.985864	0.004714	0.309639	0.109571
pooled.stratum		1	0.609386	0.060006	103.133992	0.000000	2.004237	0.197355
#INTERCEPT#		1	46.687781	2.064975	511.183583	0.000000	-	-

Variance Inflation Factors

Variable	Value
([education]=='12+ yrs')	61.837
([education]=='6-11yrs')	20.408
[parity]	5.003
[induced]	1.902
([case]==1)	1.271
[spontaneous]	2.003

Next, import the 70% linear model created earlier in R using “Insert” -> “Model from XML/PMML” and apply it to the 30% partitioned data. As before, to score the model, got to “Tools” -> “Score” to get predicted values. Now, you can compare the predicted values for the data scored in R and in KnowledgeSTUDIO. In this example, the difference between the predicted values is negligible.

4 Comparing Logistic and Linear Regression Models in KnowledgeSTUDIO and R

KnowledgeSTUDIO and R have similar performances for Logistic and Linear Regression Models, although as mentioned above, KnowledgeSTUDIO can both import and export models from/to PMML, whereas R can only export models (although not in all cases). The Logistic Regression Model in R is an extension of the General Linear Model function, thus care must be taken in the coding to specify that the analysis is Logistic. In KnowledgeSTUDIO, both the Logistic and Linear regression models can be selected as options in the Predictive Model menu. In KnoweldgeSTUDIO there are many additional options in the Regression Wizard for Logistic Regressions and Linear Regressions, which include variable selection methods and adjusting significance limits In R, the code can directly specify whether there are interactions between the variables in the model, whereas in KnowledgeSTUDIO, interactions must be specified within the data before the wizard is run. One advantage of using KnowledgeSTUDIO over R for building Logistic and Linear Regression models is that the output of KnowledgeSTUDIO displays more summary information of the model compared to R. Thus, the differences between Logistic and Linear models between R and KnowledgeSTUDIO are slight.

References

Angoss Software Corporation (2012). KnowledgeSTUDIO.

R Development Core Team (2012). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Ripley B. (2012). tree: Classification and regression trees. R package version 1.0-29. http://CRAN.R-project.org/package=tree

Therneau T. and Beth Atkinson B. R port by Brian Ripley.(2012). rpart: Recursive Partitioning. R package version 3.1-52. http://CRAN.R-project.org/package=rpart

Williams G., Hahsler M., Zementis Inc, Ishwaran H., Kogalur U. and Guha R. (2012). pmml: Generate PMML for various models. R package version 1.2.29. http://CRAN.R-project.org/package=pmml

About Angoss Software

As a global leader in predictive analytics, Angoss helps businesses increase sales and profitability, and reduce risk. Angoss helps businesses discover valuable insight and intelligence from their data while providing clear and detailed recommendations on the best and most profitable opportunities to pursue to improve sales, marketing and risk performance.

Our suite of desktop, client-server and big data analytics software products and Cloud solutions make predictive analytics accessible and easy to use for technical and business users. Many of the world's leading organizations use Angoss software products and solutions to grow revenue, increase sales productivity and improve marketing effectiveness while reducing risk and cost.

Corporate Headquarters

111 George Street, Suite 200
Toronto, Ontario M5A 2N4
Canada
Tel: 416-593-1122
Fax: 416-593-5077

European Headquarters

Surrey Technology Centre
40 Occam Road
The Surrey Research Park
Guildford, Surrey GU2 7YG
Tel: +44 (0) 1483-685-770

www.angoss.com