Predicting Breast Cancer Recurrence using Decision Trees

Abstract: A decision tree was made to classify the likelihood of breast cancer recurrence after tumour excision. The 'rpart' package was used to make the decision tree model in R. The SAS model had an accuracy of 79% and the R model had an accuracy of 73% and the number of metastatic tumours was found to be the greatest variable when classifying the likelihood of cancer recurrence.

1 Introduction

Advances in screening and treatment have allowed the 5 year survival rate for breast cancer (BC) to grow from 70% in 1980 to 85% in 2013 [1]. In the US, there are an estimated 2.8 million breast cancer survivors, accounting for 41% of all female cancer survivors [2]. For patients with low grade breast cancer, the two most common treatment options are breast conserving surgery followed by radiation therapy or mastectomy [3]. A 2010 study by Gerber et al found that 40% of BC patients suffer from a recurrence event, which required further medical intervention to treat [4]. The mortality rate of BC is increased by a factor of 2 or 3 after a recurrence event [5]. Predictive classification models for the identification of patients which are at high risk of a recurrence event may help physicians in prognosticating patient health status and in designing extended treatment plans to reduce the risk of recurrence. Decision trees are a classification method used for approximating discrete-valued target functions, in which the learned function is represented by a flowchart like structure [6]. In this experiment a decision tree model is created in SAS Enterprise Miner (EM) and R using a dataset of 286 BC patients to determine the likelihood of BC recurrence.

2 Data and Methods

The BC data set was obtained from the UCI machine learning repository. The data was donated by physicians Matjaz Zwitter & Milan Soklic from the Institute of Oncology Ljubljana, Slovenia [7]. The dataset contains 286 target instances with 9 nominal attributes (table 1). The 'class' variable was targeted for classification. The 'class' variable refers to whether or not the patient had their cancer reoccur after tumour excision. The data set contained 9 instances which had null values, since only a small number of instances contained null values each instance which contained a null value was deleted for simplicity.

Attribute	Values
class (target)	no-recurrence-events, recurrence-events
age (years)	10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90-99
menopause	lt40, ge40, premeno
tumor-size (mm)	0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59
inv-nodes (N)	0-2, 3-5, 6-8, 9-11, 12-14, 15-17, 18-20, 21-23, 24-26, 27-29, 30-32, 33-35, 36-39
node-caps	yes, no
deg-malig	1, 2, 3
breast	left, right
breast-quad	left-up, left-low, right-up, right-low, central
irradiat	yes, no

Table 1: Table showing attribute values for the BC data set.

2.1 Implementation in SAS

The process flow for the DT model in SAS EM is shown in Figure 1. The BC data was imported into SAS EM using the "File Import" node. The "Filter" node was used to remove instances which contained null values (Figure 2). The "Data Partition" node was used to partition the data into training and validation sets, the ratio was set at 80:20, respectively. No test set was used in this experiment. The "StatExplore" node was used to gain information on features of the dataset such as variable importance. Finally, the "Decision Tree" node was used to create the DT.

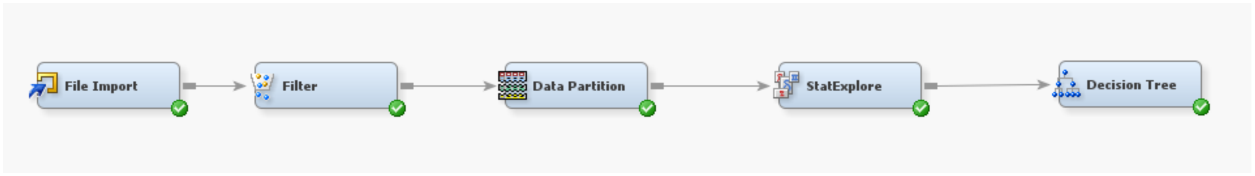
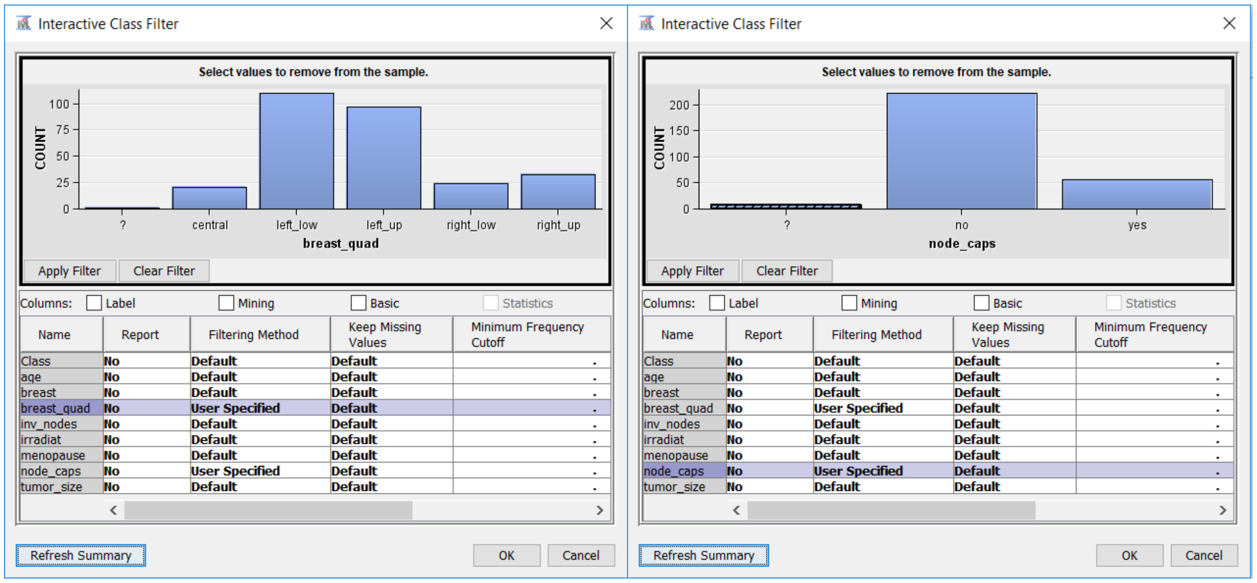


Figure 1: Process flow for DT in SAS EM



Name	Report	Filtering Method	Keep Missing Values	Minimum Frequency Cutoff
Class	No	Default	Default	.
age	No	Default	Default	.
breast	No	Default	Default	.
breast_quad	No	User Specified	Default	.
inv_nodes	No	Default	Default	.
irradiat	No	Default	Default	.
menopause	No	Default	Default	.
node_caps	No	User Specified	Default	.
tumor_size	No	Default	Default	.

Name	Report	Filtering Method	Keep Missing Values	Minimum Frequency Cutoff
Class	No	Default	Default	.
age	No	Default	Default	.
breast	No	Default	Default	.
breast_quad	No	User Specified	Default	.
inv_nodes	No	Default	Default	.
irradiat	No	Default	Default	.
menopause	No	Default	Default	.
node_caps	No	User Specified	Default	.
tumor_size	No	Default	Default	.

Figure 2: Screenshot of the interactive class filter in SAS EM. Shown in the screenshot are attributes which contained null values denoted by '?', these instances were filtered from the dataset.

2.2 Implementation in R

The BC dataset was loaded into R Studio and the instances which contained null values were omitted from the dataset. The data was partitioned with a ratio of 80:20 into training and validations sets, respectively. The 'rpart' package was used to create the decision tree and the 'rpart.plot' package was used for visualization of the tree. The general command structure of 'rpart' is:

```
rpart(formula, data, weights, subset, na.action = na.rpart, method,
model = FALSE, x = FALSE, y = TRUE, parms, control, cost, ...)
```

In this experiment the arguments (weights, subset etc.) assumed their default settings except 'data' and 'method'. Data was set to the partitioned training set (designated as train) and the method was set to class, the resultant commands take the form:

```
rpart(Class ~ .,
data = train, method = 'class')
```

Where 'Class' is the target attribute shown in table 1 [8]. The 'rpart' command uses the CART algorithm to form the decision tree. The CART decision tree is a binary recursive partitioning procedure capable of processing continuous and nominal attributes as targets and predictors [9]. In CART splitting, an instance which meets the splitting criterion is branched left and instances which do not meet the splitting criterion are branched right. At a node t, the best split s is chosen to maximise the a splitting criterion $\Delta \text{i}(s,\, t)$. When the impurity measure for a node can be defined, the splitting criterion corresponds to a decrease in impurity. The Gini impurity at a node t is defined as

$$i(t) = \Sigma_{i,j} C(i \mid j) p(i \mid t) p(j \mid t) \tag{1}$$

Where $C(i \mid j)$ is the cost of misclassifying a class j case as a class i case, $p(i \mid t)$ is the probability of case i at node t and $p(j \mid t)$ is the probability of case j at node t. In this experiment cost-sensitive learning is not used and $C(i \mid j) = 1$. The Gini splitting criterion is the decrease of impurity defined as

$$\Delta i(s,t) = i(t) - p_L \cdot i(t_L) - p_R \cdot i(t_R) \tag{2}$$

Where p_L and p_R are the respective probabilities of a case being branched to the left daughter node t_L or right daughter node t_R [10]. The DT was plotted using the 'rpart.plot' package. The plotting command 'rpart.plot' has the general structure

```
rpart.plot(x = stop("no␣'x'␣arg"),
type = 2, extra = "auto",
under = FALSE, fallen.leaves = TRUE,
digits = 2, varlen = 0, faclen = 0, roundint = TRUE,
cex = NULL, tweak = 1,
clip.facs = FALSE, clip.right.labs = TRUE,
snip = FALSE, box.palette = "auto", shadow.col = 0, ...)
```

In this experiment box.palette was set to "GnRd" and remaining arguments assumed their default settings. The resultant command takes the form:

```
 rpart.plot(mytree, box.palette="GnRd", shadow.col="grey", nn=TRUE)
```

Each node displays the classification, the probability of each class at that node and the percentage of the total instances contained at that node [11]. The nodes which were classified as non-recurrence events were set as green and the greater the likelihood of non-recurrence the darker the green. Likewise, the nodes which were classified as recurrence events were set as red and the greater the likelihood of recurrence, the darker the red.

The DT model built from the training data was tested against the the validation data using the 'predict' command. The predict command has the general structure

```
predict (object, ...)
```

The command structure was customized to classify the target of the validation data. The resultant command takes the form:

```
predict(my_tree, newdata=validate, type="class"), validate$Target)
```

2.3 Performance Metrics

A 2x2 confusion matrix of the classifications was made. Table 2 shows an example of a 2x2 confusion matrix where 'a' is the number of correct negative predictions, 'b' is the number of incorrect positive predicitions, 'c' is the number of incorrect negative predictions and 'd' is the number of correct positive predictions.

	Predicted Negative	Predicted Positive
Actual Negative	a	b
Actual Positive	c	d

Table 2: An example confusion matrix for a two-class classification problem [12]

The prediction accuracy and classification error can be calculated from the confusion matrix, they are given by:

$$Accuracy = \frac{a+d}{a+b+c+d}, \tag{3}$$

$$Error = \frac{b+c}{a+b+c+d}, \tag{4}$$

The accuracy is the proportion of the total number of predictions which were correct and the classification error is the proportion of the total number of predictions which were incorrect [12].

3 Results

DT models were created in SAS and R with the purpose of classifying a BC patient's likelihood of having their cancer recur after tumour excision, the DTs are shown in figures 6 and 7.

3.1 R Results

A DT was made using the BC dataset in R (Figure 5). The DT model created in R using the training data was tested against the validation data, Table 1 shows the confusion matrix. The classification accuracy for the model was 0.7346939 (Equation 3) and the classification error was 0.1842105 (Equation 4).

	no-recurrence-events	recurrence-events
no-recurrence-events	31	7
recurrence-events	6	5

Table 3: Confusion matrix for model tested against the validation set

The printcp() command was used to gain information on the complexity parameter (cp) and the cross validation error was plotted as a function of cp and tree size (Figure 3).

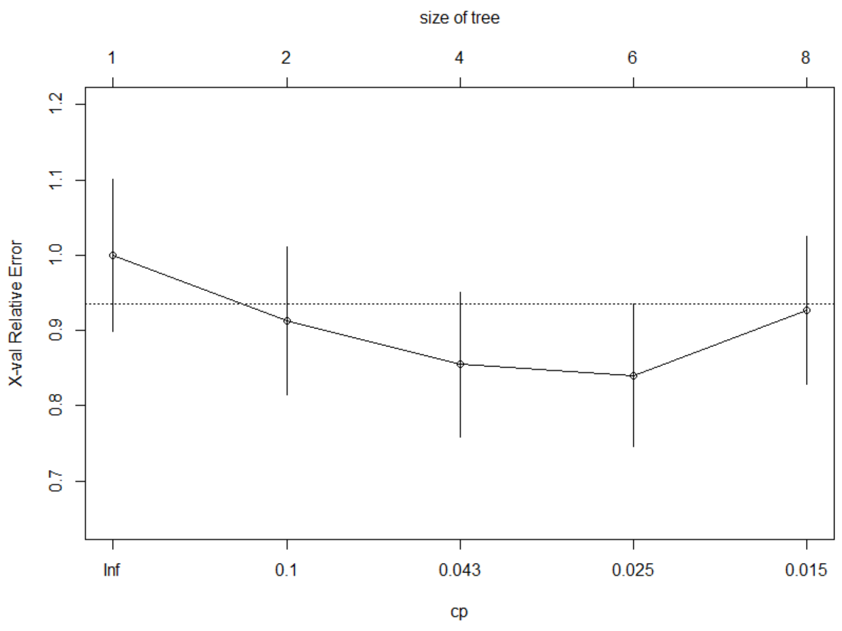


Figure 3: Cross validation error (X-val Relative Error) as a function of cp and tree size. All variables have no units.

3.2 SAS Results

The StatExplore node was used to produce a statistical summary of the input data. Figure 4 shows variable importance for the DT, 'inv-nodes' was the most important variable when classifying BC recurrence and 'breast' was the least important variable.

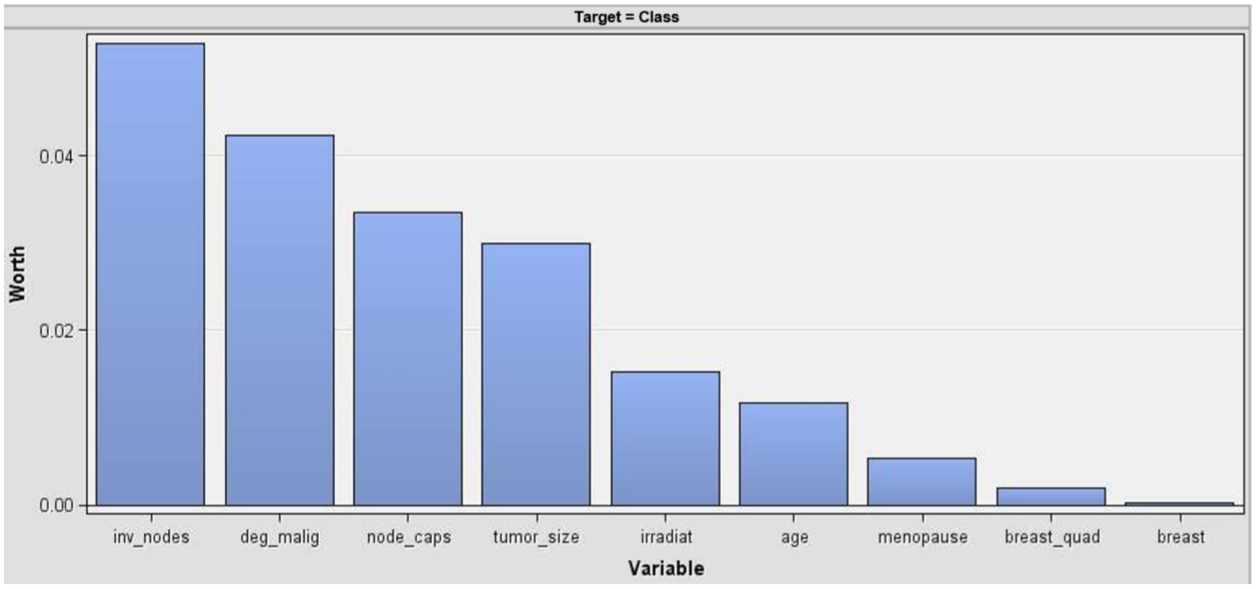


Figure 4: Plot showing variable importance for each attribute in the model.

The Leaf Statistics window from the Decision Tree node plots the percentage of each observation in each leaf node. Figure 5 shows the percentage of each observation in each leaf node for the training and validation sets. The most common level in a node is the predicted value assigned to that node. The difference in size between the training and validation bars gives the error. Figure 5 shows that leaf 1, 2 and 5 have high errors.

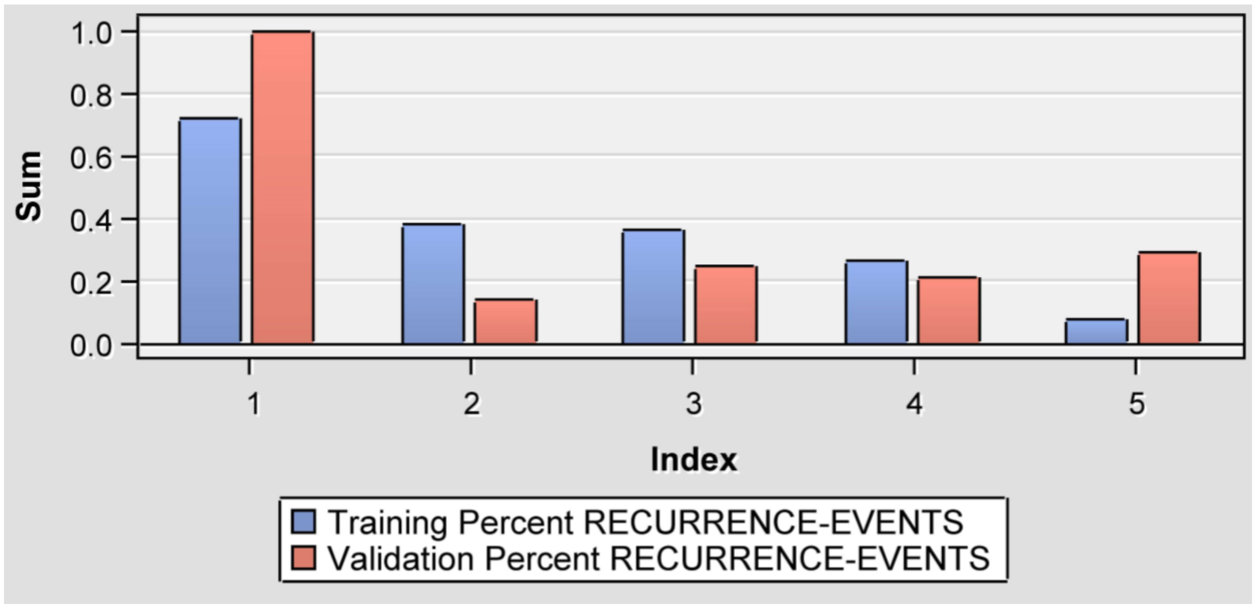
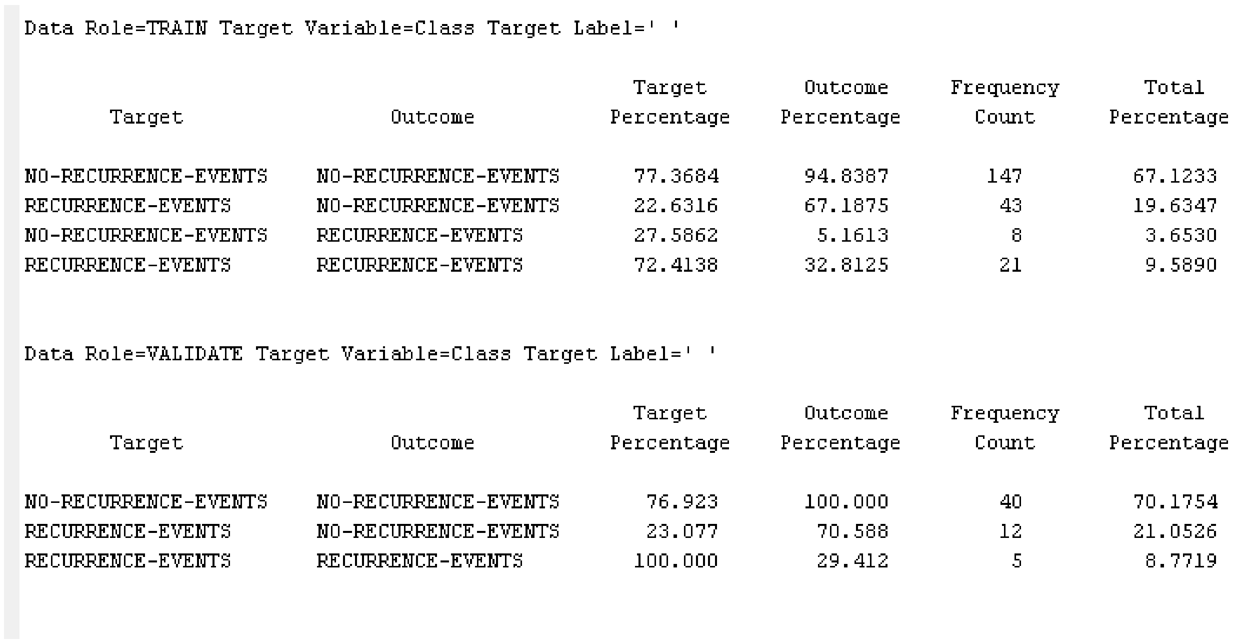


Figure 5: The percentage of each observation in each leaf node for the training and validation sets



```
Data Role=TRAIN Target Variable=Class Target Label=' '

                                                  Target      Outcome    Frequency      Total
       Target                   Outcome         Percentage  Percentage     Count      Percentage

NO-RECURRENCE-EVENTS    NO-RECURRENCE-EVENTS     77.3684     94.8387        147        67.1233
RECURRENCE-EVENTS       NO-RECURRENCE-EVENTS     22.6316     67.1875         43        19.6347
NO-RECURRENCE-EVENTS    RECURRENCE-EVENTS        27.5862      5.1613          8         3.6530
RECURRENCE-EVENTS       RECURRENCE-EVENTS        72.4138     32.8125         21         9.5890


Data Role=VALIDATE Target Variable=Class Target Label=' '

                                                  Target      Outcome    Frequency      Total
       Target                   Outcome         Percentage  Percentage     Count      Percentage

NO-RECURRENCE-EVENTS    NO-RECURRENCE-EVENTS      76.923     100.000         40        70.1754
RECURRENCE-EVENTS       NO-RECURRENCE-EVENTS      23.077      70.588         12        21.0526
RECURRENCE-EVENTS       RECURRENCE-EVENTS        100.000      29.412          5         8.7719
```

Figure 6: Classification data from the Decision Tree node output

	Predicted Negative	Predicted Positive
Actual Negative	40	0
Actual Positive	12	5

Table 4: Confusion matrix formed withthe validation data from the classification table in SAS

A confusion matrix was constructed using the classification data shown in figure 6. Table 4 shows the confusion matrix, using equations 3 and 4 we find that Accuracy = 0.78947 to 5 s.f and Error = 0.21052 to 5 s.f

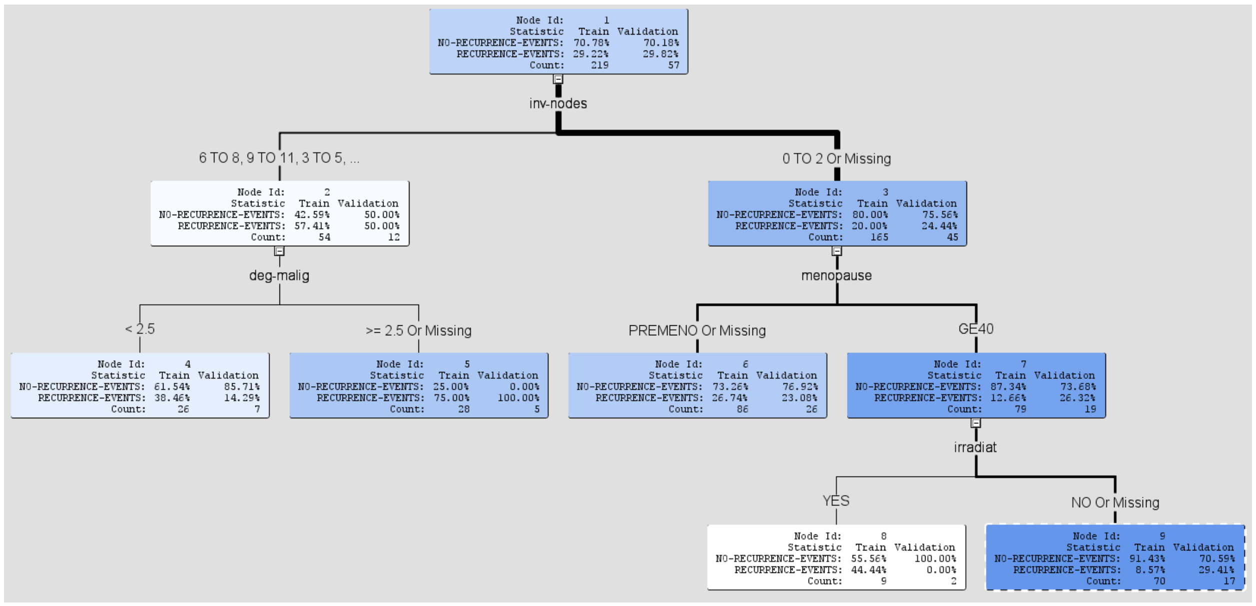

Node Id: 1
Statistic Train Validation
NO-RECURRENCE-EVENTS: 70.78% 70.18%
RECURRENCE-EVENTS: 29.22% 29.82%
Count: 219 57
inv-nodes
6 TO 8, 9 TO 11, 3 TO 5, ...
0 TO 2 Or Missing
Node Id: 2
Statistic Train Validation
NO-RECURRENCE-EVENTS: 42.59% 50.00%
RECURRENCE-EVENTS: 57.41% 50.00%
Count: 54 12
Node Id: 3
Statistic Train Validation
NO-RECURRENCE-EVENTS: 80.00% 75.56%
RECURRENCE-EVENTS: 20.00% 24.44%
Count: 165 45
deg-malig
menopause
< 2.5
>= 2.5 Or Missing
PREMENO Or Missing
GE40
Node Id: 4
Statistic Train Validation
NO-RECURRENCE-EVENTS: 61.54% 85.71%
RECURRENCE-EVENTS: 38.46% 14.29%
Count: 26 7
Node Id: 5
Statistic Train Validation
NO-RECURRENCE-EVENTS: 25.00% 0.00%
RECURRENCE-EVENTS: 75.00% 100.00%
Count: 28 5
Node Id: 6
Statistic Train Validation
NO-RECURRENCE-EVENTS: 73.26% 76.92%
RECURRENCE-EVENTS: 26.74% 23.08%
Count: 86 26
Node Id: 7
Statistic Train Validation
NO-RECURRENCE-EVENTS: 87.34% 73.68%
RECURRENCE-EVENTS: 12.66% 26.32%
Count: 79 19
irradiat
YES
NO Or Missing
Node Id: 8
Statistic Train Validation
NO-RECURRENCE-EVENTS: 55.56% 100.00%
RECURRENCE-EVENTS: 44.44% 0.00%
Count: 9 2
Node Id: 9
Statistic Train Validation
NO-RECURRENCE-EVENTS: 91.43% 70.59%
RECURRENCE-EVENTS: 8.57% 29.41%
Count: 70 17

Figure 7: DT created in SAS Enterprise Miner

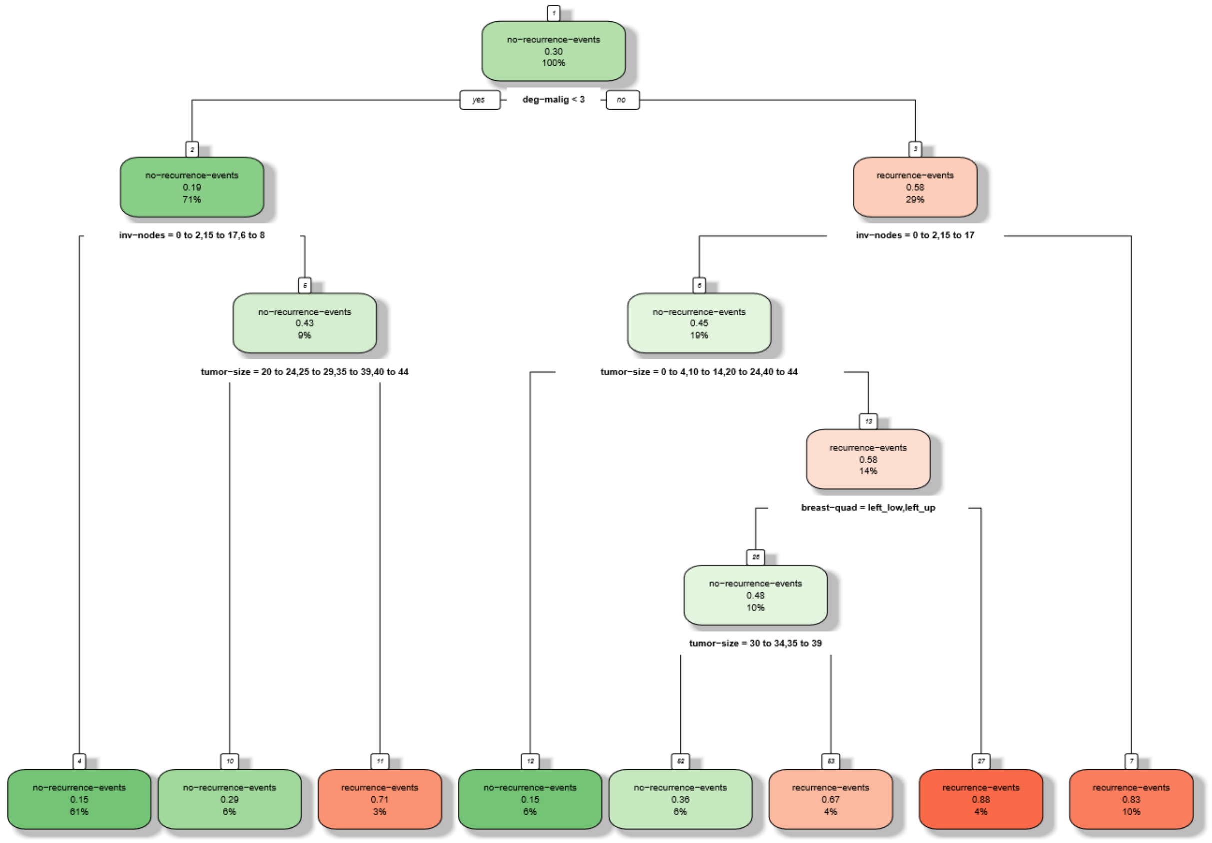


Figure 8: DT created in R using 'rpart'

4 Discussion

The BC dataset contained a small number of instances (286) which gives it lower population validity. Trees built from a small sample are less accurate than trees built from a large sample, so future research should make use of a larger dataset [13] In this report accuracy and error derived from confusion matrices were used as a performance metric for DT models. Accuracy and error are easy to compute and easy to understand but are limited because they produce less distinctive and less discriminable values, this leads to less ability to select the optimal classifier [14]. Furthermore, this experiment was limited by the fact that cost-sensitive learning was not used, so the accuracy and error derived would be based on equal cost for each type of error which occurs. Future research could make use of cost-sensitive learning to improve the precision of the performance metric.

	R	Sas
Classification Accuracy	73%	79%
Classification Error	18%	21%

Table 5: Table comparing classification accuracy and error for tree models in R and SAS

Table 5 shows a comparison of model accuracy in SAS and R. SAS provided a more accurate model (79%) than R (73%). The SAS DT was automatically pruned whereas the R DT was not pruned. Pruning is a technique which increases the generalizability of a dataset pruning by lowering the risk of overfitting. The R tree was not pruned since all of the branch levels were consistent with one another within one standard error (Figure 3).

5 Conclusions

Predictive classification models for the identification of patients which are at high risk of a BC recurrence event may help physicians in prognosticating patient health status and in designing extended treatment plans to reduce the risk of recurrence. In this report a DT model was created in R and SAS EM. The 'rpart' package was used to create the model in R, 'rpart' uses the CART algorithm to make the DT. The CART splitting procedure is a binary recursive partitioning procedure capable of processing continuous and nominal attributes as targets and predictors. Confusion matrices were used a performance metric for each model. The SAS model had an accuracy of 79% whereas the R model had an accuracy of 73%. The R model was not pruned because all levels of branch splitting were consistent within one standard error.

References

[1] A. C. Society, *Cancer Facts & Figures 2017-18.* Atlanta American Cancer Society, Inc., 2017.

[2] B. Ansa, W. Yoo, M. Whitehead, S. Coughlin, and S. Smith, "Beliefs and behaviors about breast cancer recurrence risk reduction among african american breast cancer survivors," *International journal of environmental research and public health*, vol. 13, no. 1, p. 46, 2015.

[3] A. Ahmad, "Pathways to breast cancer recurrence," *ISRN oncology*, vol. 2013, 2013.

[4] B. Gerber, M. Freund, and T. Reimer, "Recurrent breast cancer: treatment strategies for maintaining and prolonging good quality of life," *Deutsches Arzteblatt international*, vol. 107, no. 6, p. 85, 2010.

[5] R. Dent, A. Valentini, W. Hanna, E. Rawlinson, E. Rakovitch, P. Sun, and S. Narod, "Factors associated with breast cancer mortality after local recurrence," *Current Oncology*, vol. 21, no. 3, p. e418, 2014.

[6] T. M. Mitchell *et al.*, "Machine learning," *Burr Ridge, IL: McGraw Hill*, vol. 45, no. 37, pp. 52–59, 1997.

[7] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[8] T. Therneau, B. Atkinson, B. Ripley, and M. B. Ripley, "Package 'rpart'," *Available online: cran. ma. ic. ac. uk/web/packages/rpart/rpart. pdf (accessed on 5 December 2018)*, 2018.

[9] D. Steinberg and P. Colla, "Cart: classification and regression trees," *The top ten algorithms in data mining*, vol. 9, p. 179, 2009.

[10] L. Breiman, *Classification and regression trees.* Routledge, 2017.

[11] S. Milborrow, "Plot 'rpart'models: an enhanced version of 'plot. rpart'," *R package version*, vol. 2, no. 1, 2016.

[12] S. Visa, B. Ramsay, A. L. Ralescu, and E. Van Der Knaap, "Confusion matrix-based feature selection." in *MAICS*, 2011, pp. 120–127.

[13] J. Catlett, "Overprvning large decision trees." in *IJCAI*, 1991, pp. 764–769.

[14] M. Hossin and M. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, p. 1, 2015.

A R Script

```
#### Load data
setwd("")
breast_data <- read.csv("newtest␣-␣Copy.csv", header= TRUE)

#### Inspect data
str(breast_data); class(breast_data); mode(breast_data)
apply(is.na(breast_data),2,sum) # checks for null values in data set, none
   ↪ are present
names(breast_data); head(breast_data); tail(breast_data);summary(breast_data
   ↪ )
nrow(breast_data); ncol(breast_data); dim(breast_data)

#### Rename variables
find.package('plyr')
library(plyr)
renam <- data.frame(breast_data, check.names = TRUE) #Variables will need
   ↪ to renamed for clarity
names(renam) #Check for variable names
rename(renam, c('Column1'='Class','Column2'='age', 'Column3'='menopause','
   ↪ Column4'='tumor-size','Column5'='inv-nodes','Column6' = 'node-caps',
'Column7'='deg-malig', 'Column8'= 'breast', 'Column9'='breast-quad', '
   ↪ Column10'='irradiat' ))
count(renam == '?') #Unknown values were recorded as '?' in the data set.
   ↪ 9 unknown values are present. Rows containing '?' are removed for
   ↪ simplicity
breast_tree <- rename(renam, c('Column1'='Class','Column2'='age', 'Column3'=
   ↪ 'menopause','Column4'='tumor-size','Column5'='inv-nodes','Column6' =
   ↪ 'node-caps',
'Column7'='deg-malig', 'Column8'= 'breast', 'Column9'='breast-quad', '
   ↪ Column10'='irradiat' ))

which(breast_tree == '?', arr.ind = TRUE) # col 6 & 9 contain '?'
names(breast_tree) # col 6 = node-caps, col 9 = breast-quad
cleantree <- breast_tree[breast_tree$`node-caps`!='?' & breast_tree$`breast-
   ↪ quad`!='?',] # omit rows containing missing values
dim(cleantree) # rows omitted

#### Decision Tree Packages
find.package(rpart)
find.package((rpart.plot))
install.packages('rpart')
install.packages('rpart.plot')
```

```
library(rpart)
library(rpart.plot)

#### Train tree
set.seed(1234) #seed for training set
pd <- sample(2, nrow(cleantree),replace=TRUE, prob=c(0.8,0.2))
pd
train <- cleantree[pd==1,]
validate <- cleantree[pd==2,]
cfit <- rpart(Class ~ .,
data = train, method = 'class')
print(cfit)
rpart.plot(cfit, box.palette="GnRd", shadow.col="grey", nn=TRUE)

#### Test tree
pred <- predict(cfit, train, type="class") #designates predict(cfit) as '
   ↪ pred'
tab <- table(pred, train$Class)
test_predict <- table(predict(cfit, newdata=validate, type="class"),
   ↪ validate$Class)
sum(diag(test_predict))/(sum(test_predict)) # classification accuracy
1-sum(diag(tab))/sum(tab) # classification error

#### Post pruning
printcp(cfit); plotcp(cfit)
cp_table <- printcp(cfit)
cp_table2 <- table(cp_table)
which.min(cp_table2)
prune_node <- cp_table[which.min(cp_table[,"xerror"]),"CP"]
prune_tree <- prune.rpart(cfit, cp=prune_node)
rpart.plot(prune_tree, box.palette="GnRd", shadow.col="grey", nn=TRUE)
printcp(prune_tree)
plotcp(prune_tree) # visualize error as size of tree grows
rpart.plot(cfit, box.palette="GnRd", shadow.col="grey", nn=TRUE)
pred <- predict(cfit, train, type="class") #designates predict(cfit) as '
   ↪ pred'
tab <- table(pred, train$Class)
dim(pred); dim(train)
print(tab)
sum(diag(tab))/sum(tab)
test_predict <- table(predict(cfit, validate, type="class"), validate$Class)
print(test_predict)
a <- sum(diag(test_predict))/(sum(test_predict)) # classification accuracy
b <- 1-sum(diag(tab))/sum(tab) # classification error
pred_prune <- predict(prune_tree, train, type="class")
```

```
tab_prune <- table(pred_prune, train$Class)
print(tab_prune)
sum(diag(tab_prune))/sum(tab_prune)
test_prune <- table(predict(prune_tree, validate, type="class"), validate$
   ↪ Class)
sum(diag(test_prune))/(sum(test_prune))
1-sum(diag(tab_prune))/sum(tab_prune)
```