Data Mining : Wisconsin Prognostic Breast Cancer Dataset

Joiret Marc (1541822)

March 30, 2017

1 Introduction

In this Data Mining project, a number of supervised and unsupervised machine learning techniques are used on a real dataset for comparison purposes. The usefulness and prediction accuracy of a particular data mining technique is most often dataset dependent. Two learning problems are addressed in this project : predicting a continuous outcome and classifying a categorical (binary) outcome in the dataset from a number of $p > 10$ predictor covariates (which can be continuous or categorical). Three data mining tasks are conducted :

1. Supervised learning to predict a continuous outcome as precisely as possible. Three approaches are compared : Lasso regression (parametric shrinkage method), regression tree (non parametric method) and random forest (non parametric).
2. Supervised learning to classify a binary outcome as accurately as possible (with lowest misclassification rate as possible). Two approaches are compared : K-nearest neighbors (non parametric) and Support Vector Machine.
3. Unsupervised learning to discover patterns or clusters (stratification or segmentation in the data). Hierarchical and Partition Clustering are compared.

1.1 Dataset

The approved dataset was described in a previous document.

There are 198 observations (patients) with 31 input variables, one continuous output variable (tumor size) and one binary outcome variable (recurrent or non recurrent tumor). 151 patients in the dataset had a non-reccurent tumor (76.26%) and 47 patients had a recurrent tumor (23.74%).

The interested reader will find a full description in the appendix.

1.2 Objectives of the Study

The primary objective of the study is to develop a classifier for predicting whether a patient has a recurrent or a non-recurrent breast cancer from the characteristics of a fine needle tissue aspiration (a tissue sample taken with a hollow needle from a mass under the skin), plus Tumor Size and Lymph Node Status at Surgery time. A secondary objective is to predict the size of the tumor (in centimeters) and compare it with the real excised tumor size. See Mangasarian and Wolberg for details [2].

1.3 Missing Data or Missing Attributes

One of the attribute in the data : ”Lymph Node Status” (the number of positive axillary lymph nodes observed at time of surgery) is missing for 4 patients (patients ID = 844359, 854253, 877500 and 947204). Examining Figure 7 in appendix, it is suspected that the Lymph Node Status is partially informative in classifying between Recurrent (blue) or Non recurrent (red) breast cancer.

Non-recurrent tumors tend to be associated with a higher occurrence of Lymph Node Status equal to zero. For this reason, the attribute was kept but the 4 patients with this missing attribute were removed from the data. Removing 4 patients out of the 198 with their 32 variables is better than removing the whole attribute for 194 patients because less information is lost.

The assumption is made that the attribute was missing completely at random across the patients. The fraction of observations removed is about 2% and is considered acceptable.

We kept all the remaining 194 patients in the analysis and most importantly kept all the attributes.

2 Methods

The used data mining methods are described hereafter and the corresponding R scripts are given in the appendix.

2.1 Supervised Learning for prediction

2.1.1 Lasso Regression (parametric)

For prediction accuracy and model interpretability, a linear model with regularization (shrinkage) method is used to predict the Tumor Size (in centimeters) continuous outcome at time of surgery. The Lasso[1] modeling approach involves fitting a linear model with all the p=31 predictors but with a shrinkage performed so that some of the coefficients may be estimated exactly to zero. Hence, this approach can also perform variable selection and is prone to deliver a model that is easier to interpret. Constraining or shrinking the estimated coefficients can reduce the variance at the cost of some increase in bias (see [4] - Chap. 6).

Lasso Model : The lasso coefficients β^L_λ minimize the residual sum of square (RSS) as in classical linear regression but with the extra constraint that the sum of the absolute values of the β_j be less than some given constraint parameter t.

$$\beta_\lambda^{Lasso} = \text{argmin}_\beta \sum_{i=1}^{n} \big(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\big)^2$$

$$\text{subject to} \sum_{j=1}^{p} |\beta_j| \le t$$

Minimizing a function under a constraint is equivalent to the Lagrange formulation to solve an optimization problem :

$$\beta_\lambda^{Lasso} = \text{argmin}_\beta \Big\{ \frac{1}{2} \sum_{i=1}^{n} \big(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\big)^2 + \underbrace{\lambda \sum_{j=1}^{p} |\beta_j|}_{L_1 \text{ penalty}} \Big\}$$

Tuning : The λ coefficient in the penalty term is used as tuning parameter to minimize the mean square error by cross validation on the training set. The minimal Mean Square Error (MSE) at optimum λ is computed. Once the optimum λ is found, the fitted Lasso model is used for prediction on the test set.

Standardization : The variables in the data are standardized so that they are on the same scale prior to fitting the model sequence (the `glmnet()` function in the R `glmnet` library does standardization by default and returns the coefficients back on the original scale [3]).

[1]LASSO is an acronym for Least Absolute Shrinkage and Selection Operator.

Precision Measure of the prediction : The Lasso model predicted values for the Tumor Size based on the predictor values of the test dataset are compared with the test real observed Tumor Size outcomes. A Mean square error is computed (and standard deviation) that will provide the measure of the prediction precision.

2.1.2 Regression Tree (non parametric)

A non parametric tree based model is chosen for the continuous Tumor Size outcome prediction, namely the CART method (Classification and Regression Tree). The algorithm is the following ([4] - Chap. 8, pp.303-311) :

Tree Growing : Use top-down recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has few than some minimal number of observations.

Tree Pruning : The criterion for a good tree is the one with the lowest test error rate. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtree as a function of the tuning parameterα. For each value of α there is a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2 + \alpha|T|$$

where T indicates the number of terminal nodes of the tree and $\hat{y}_{R_m}$ is the mean of the training observations in the R_m. The tuning parameter α controls the trade-off between the tree complexity and its fit to the training data.

K-fold cross-validation : The optimum α is found by K fold cross-validation. That is, the training data is divided into K folds. For each $k = 1, \cdots, K$:

1. Repeat Tree Growing and Tree Pruning on all but the k^{th} fold of the training data.
2. Evaluate the mean squared prediction error (deviance) on the data in the left-out k^{th} fold as a function of α.

Average the results for each value of α and pick up the one that minimizes the error.

Final subtree selection : Return the subtree from the Tree Pruning that corresponds to the optimal α.

2.1.3 Random Forest (non parametric)

The decision trees suffer from high variance. Bootstrap aggregation (or bagging) is a general purpose procedure for reducing the variance of statistical learning methods. Many training sets replicates are generated by bootstrapping (sampling with replacement) within the single training set. We finally average all the predictions for all the bootstrap replicates. This is called bagging. There is no single tree anymore easy to interpret but the relative importance of the variables can be ranked for RSS or node purity reduction. Random forests provide an improvement over bagged trees by way of de-correlating the trees (see [4]-Chap. 8.2.2, pp.320-321). Each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of the p predictors. The split is allowed to use only one of the m predictors. A new sample of m predictors is taken at each split, and typically $m = \sqrt{p}$ (in our data, m=6 predictors are randomly selected at each split). This prevents the shortcoming observed in bagging that all the trees look quite similar to each other if there is some strong predictor in the dataset which will be used in the top split. Random forests overcome this problem by forcing each split to consider only a subset of predictors. Using a small value of m in building a random forest is typically helpful when there is a large number of correlated predictors and in high dimensional data.

2.2 Supervised Learning for classification

Two classification approaches will be compared on the breast cancer dataset : k-Nearest Neighbors and Support Vector Machine with radial kernel.
The model accuracy is assessed by the classification accuracy (or equivalently by the misclassification error rate).
Depending on the meaning of the predicted outcome, a full confusion matrix, or a sensitivity (True Positive Rate), or a specificity (1 minus False Positive Rate), or a ROC curve, can be used to assess the predictive performance of a classifier.
In the breast cancer dataset application, the sensitivity is paramount because it is more important for a recurrent patient to be correctly classified than for a non recurrent patient. Hence, accuracy, sensitivity and specificity will be compared for all the classifiers but with sensitivity being the primary concern here for the quality of the classifier.

2.2.1 k-nearest-neighbors - kNN (non parametric)

The k-Nearest Neighbors approach is used to classify the type of tumor in the breast cancer dataset.
The k-Nearest Neighbors classifier takes a positive integer k and a test observation x_0, identifies the k points in the training data that are closest to x_0 (using some distance metric like the euclidean distance). The closest points belong to a subset called $\mathcal{N}_0$. Then, the conditional probability for class j is estimated as the fraction of points in $\mathcal{N}_0$ whose response values equal j :

$$P(Y = j|X = x_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

Finally, the kNN method classifies the test observation x_0 to the class with the largest probability.
It is important to standardize the data (scaling) before conducting the kNN classification so that there is no scale effect due to differences in variable scales (differences in measurement units).
The precision of the classifier depends on k. With k=1, the training error rate is low, but the test error rate may be quite high. As $1/k$ increases the method is more flexible but the error rate gets worse. Cross validation will be used to find the optimal k for which the error rate is minimum or for which the sensitivity is the largest.

2.2.2 Support Vector Machine with radial kernel

The Support Vector Machine (SVM) extends the support vector classifier using kernels in order to enlarge the features space to accommodate a non-linear boundary between the classes [4] - Chap.9.3, pp.350-366. SVM algorithms use inner products of the observations instead of the observations themselves. A generalization replaces the proper inner product by a kernel function of different non linear forms like the radial kernel, which writes

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

Note that for large Euclidean distances, the kernel is small (the data should also be standardized). The support vector classifier, when non linear, is called a support vector machine and has the form

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$$

The radial basis function maps samples into a higher dimensional space (the kernel trick) and can handle relations between class labels and predictors that are non linear.

SVM Tuning : When fitting an SVM with a radial kernel, two parameters can affect the results : γ and cost. γ is the kernel parameter that controls the shape of the separating hyperplane. Larger values for γ results in larger number of support vectors. The cost parameter represents the cost of making errors. A large value severely penalizes errors and yields more complex classification boundaries, less misclassification, but overfitting may result in poor predictive power in new samples. γ and cost are always positive real values. Different combination of gamma and cost are tuned via a grid search ($\gamma = 0.1, 0.5, 1.5, 2, 3, 5$ and cost=1, 10, 100, 1000). The best combination with fewest errors is retained by cross-validation.

2.3 Unsupervised Learning

The questions we are addressing in unsupervised analysis with the breast cancer dataset are :

1. What are the similarities and differences among the 194 patients based on the 32 measures made on their fine needle aspirate, tumor size and lymph node status ?
2. Is there a pattern or a smaller number of groups into which these patients can be meaningfully clustered ? How many subtypes are there (if any) and what are their characteristics ? We did not incorporate the known binary outcome (R or N) in the analysis because a genuinely unsupervised analysis is conducted.

Finally, with the partition clustering around medoids (with k=2) we can assess the meaning of our obtained clusters and see if there is some association with the observed binary classification (this could be viewed as a supervised meaningful test of the PAM clustering results).
Again, in order to prevent units scale effects, we standardized the data (mean zero and unit standard deviation) before conducting the analysis.

2.3.1 Hierarchical Cluster Analysis

The Hierarchical clustering does not require to choose the number of clusters upfront. We use here the agglomerative (bottom-up) as opposed to the divisive paradigm (top-down). The algorithm is as follows

1. Lowest level start : Define each observation as a cluster (row).
2. Distance calculation : Calculate the distances (single linkage, complete linkage, average linkage, centroid or Ward) between every cluster and every other cluster.
3. Merging : Combine the two clusters that have the smallest distance. This reduces the number of clusters by one. The average linkage distance is used in our analysis because it is less likely to chain and less susceptible to outliers.
4. Repeat steps 2 and 3 : until all clusters have been merged into a single cluster containing all the observations.

Rather than performing hierarchical clustering on the entire data matrix, it can be performed on the first few principal component score vectors. This helps to remove a bit of noise in the data. We performed the hierarchical cluster analysis on the first 4 PC of the 32 predictors.

2.3.2 Partitioning Cluster Analysis

Partitioning Around Medoids (PAM) is used because it is more robust to outliers than the k-means cluster analysis. Rather than representing each cluster using a centroid (a vector of variable means), each cluster is identified by its most representative observation (called a medoid). Any distance measurement can be used but the Euclidean metric distance is used here. The PAM algorithm is as follows [6] - Chap 16, pp.382-383 :

1. Randomly select K observations (call each a medoid).
2. Calculate the distance/dissimilarity of every observation to each medoid.
3. Assign each observation to its closest medoid.
4. Calculate the sum of the distances of each observation from its medoid (total cost)

5. Select a point that isn't a medoid, and swap it with its medoid.
6. Reassign every point to its closest medoid.
7. Calculate the total cost.
8. If this total cost is smaller, keep the new point as a medoid.
9. Repeat steps 5-8 until the medoids don't change.

3 Results

3.1 Supervised Learning for prediction

3.1.1 Lasso Regression results

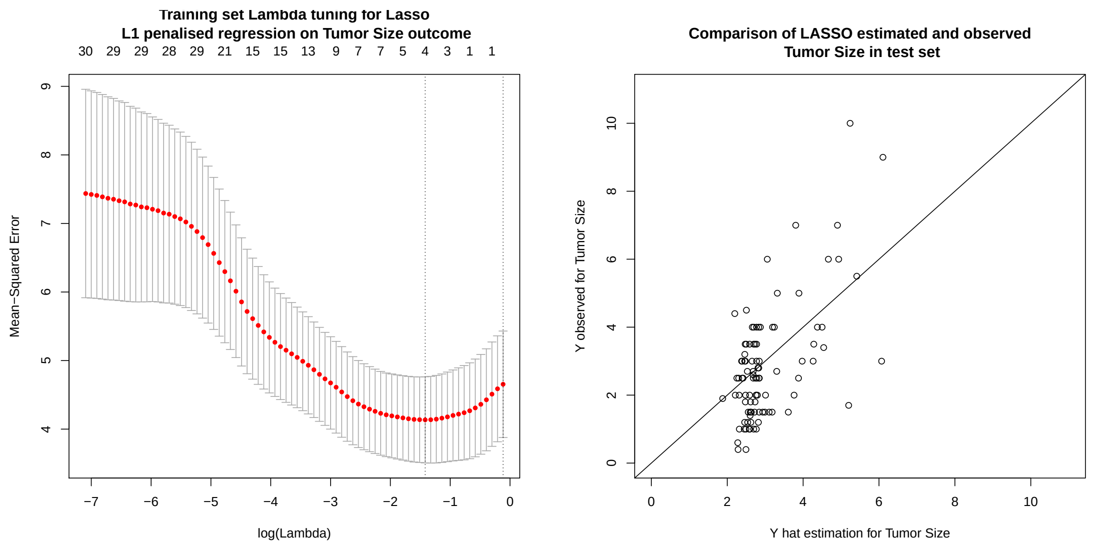

Figure 1: Lasso λ tuning on the training set minimizing Mean Square Error (left panel) and Comparison of estimated and observed Tumor Size on the test set (right panel).

The best lambda value obtained by tuning is $\lambda = 0.2420956$ ($log\lambda = -1.418423$) providing the lowest error on the training set and retaining 4 parameters, including the intercept (see left panel on Figure 1). The best Lasso model to predict the Tumor Size reads

$$Y = \beta_0 + \beta_7 \cdot \text{Area Mean} + \beta_{12} \cdot \text{Symmetry Mean} + \beta_{35} \cdot \text{Lymph Node Status} \tag{1}$$

where the β's are tabulated in Table 1.

Table 1: Fitted Lasso model estimated parameters with tuning λ minimizing MSE.

Effect	Parameter	Estimated value
Intercept	β_0	2.557497
Area Mean	β_7	0.000127
Symmetry Mean	β_{12}	-1.171983
Lymph Node Status	β_{35}	0.128490

The right panel in Figure 1 compares the predicted and the observed Tumor Size on the test set. The Mean Square Error and standard deviation ($\widehat{s.e.}$) on the Tumor Size Prediction based on the Lasso model on the test dataset are MSE$= 1.916$ and $\widehat{s.e.} = 1.384$ centimeter.

3.1.2 CART results

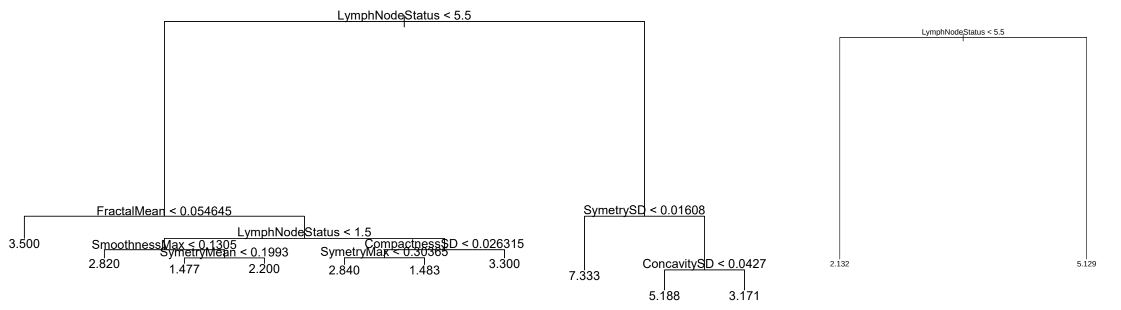

Figure 2: Classification and Regression Tree for the Tumor Size in cm (left panel : initial tree, right panel : final tree).

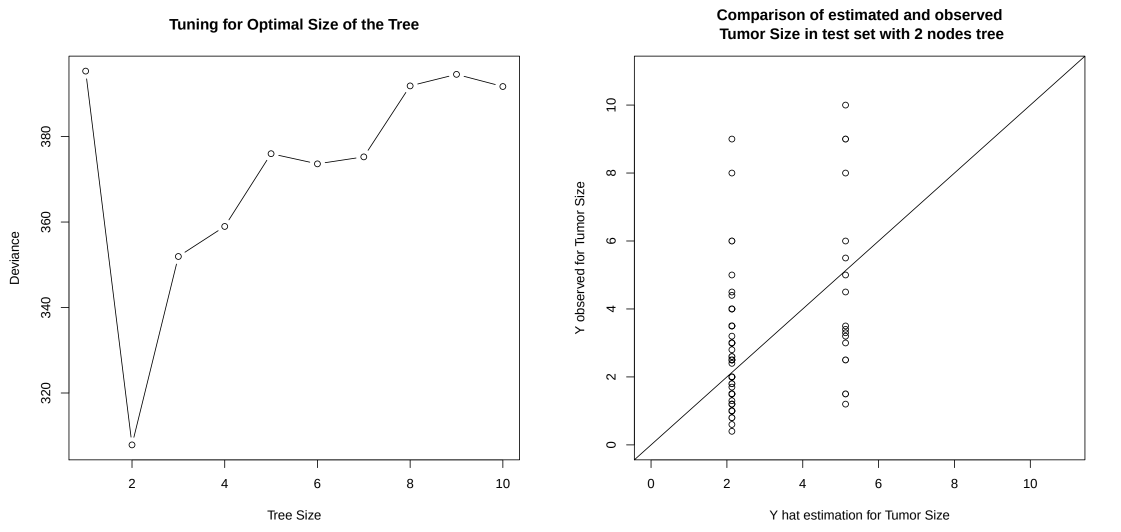

Figure 3: Tree size tuning (left panel) and Final Regression Tree (with 2 nodes) Model Evaluation (right panel).

The best Classification and Regression Tree (CART) model is very simple and is graphically displayed on the right panel of Figure 2. The CART prediction reads like this : if the Lymph Node Status is less than or equal to 5, the predicted Tumor Size is 2.132 cm, otherwise the predicted Tumor Size is 5.129 cm.
The right panel in Figure 3 compares the predicted and the observed Tumor Size on the test set.
The Mean Square Error and standard deviation $(\widehat{s.e.})$ on the Tumor Size Prediction based on the CART model on the test dataset are MSE$= 3.466$ and $\widehat{s.e.} = 1.862$ centimeter.

3.1.3 Random Forest results

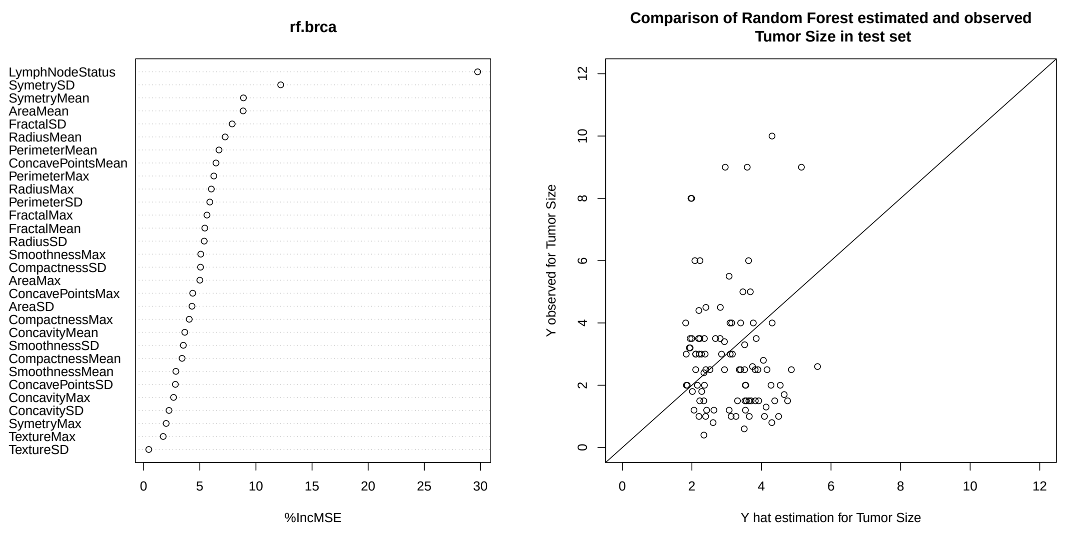

Figure 4: Random Forest Variable Importance for Tumor Size (left panel) and Comparison of estimated and observed Tumor Size on the test set (right panel).

The 4 most important variables, causing most of the decrease in accuracy in the out-of-bag sample when excluded, for the Tumor Size prediction are Lymph Node Status, Symmetry SD, Symmetry Mean and Area Mean. The right panel in Figure 4 compares the predicted and the observed Tumor Size on the test set. The Mean Square Error and standard deviation ($\widehat{s.e.}$) on the Tumor Size Prediction based on Random Forest model (with m=6 predictors at each split) on the test dataset are MSE= 4.806 and $\widehat{s.e.} = 2.192$ centimeter.

4 Supervised Learning for Classification

4.0.1 k-nearest-neighbors - kNN results

Figure 5 shows the k-NN test error rate (in black) along with the sensitivity, i.e the True Positive Rate (in red), as the level of flexibility (assessed using $1/k$) increases or equivalently as the number of neighbors decreases. A good compromise between accuracy and specificity is obtained when k=3. A value of k=3 gives the best sensitivity ($= 12.9\%$) on the test set, which is a poor performance for a diagnostic tool. The confusion matrix for the k-NN classifier with k=3 is tabulated in Table 2.

Table 2: Confusion matrix for the kNN classifier with k=3 on the test set.

		Predicted class		
		− or Non recurrent	+ or Recurrent	Total
True	- or Non-recurrent	tn=61	fp=5	66
class	+ or Recurrent	fn=27	tp=4	31
	Total	88	9	97

The accuracy is $(\text{tn} + \text{tp})/97 = (61 + 4)/97 = 0.67$. The misclassification rate is $1 - 0.67 = 0.33$. The sensitivity (True Positive Rate) is $\text{tp}/(\text{fn} + \text{tp}) = 4/(27 + 4) = 0.129$. The specificity (1 minus False Positive Rate) is $\text{tn}/(\text{tn} + \text{fp}) = 61/(61 + 5) = 0.924$. Although the accuracy is good, the sensitivity (=power) of the kNN classifier is poor and the type II error (probability of accepting the null given it is false) is 0.871.

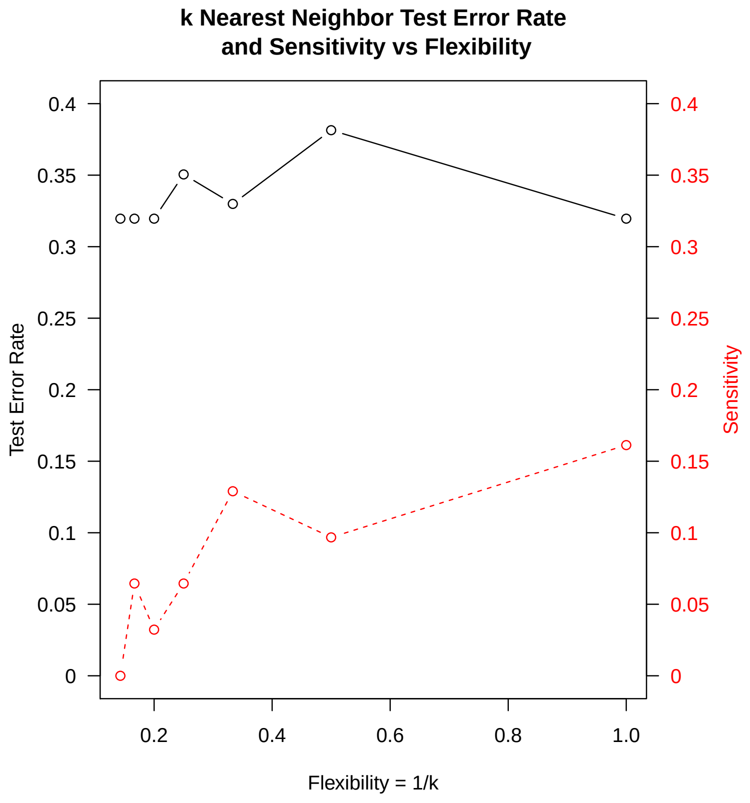

Figure 5: Misclassification rate (black) and sensitivity (red) versus k=number of nearest neighbors.

4.0.2 Support Vector Machine with radial kernel results

The best tuned parameters are $\gamma = 0.1$ and cost = 1.
The confusion matrix for the SVM classifier with $\gamma = 1$ and cost=1 for the test set is tabulated in Table 3.

Table 3: Confusion matrix for the SVM classifier with $\gamma = 0.1$ and cost=1 on the test set.

		Predicted class		
		$-$ or Non recurrent	+ or Recurrent	Total
True class	- or Non-recurrent	tn=59	fp=7	66
	+ or Recurrent	fn=28	tp=3	31
	Total	87	10	97

The accuracy is $(\text{tn} + \text{tp})/97 = (59 + 3)/97 = 0.64$. The misclassification rate is $1 - 0.64 = 0.36$. The sensitivity (True Positive Rate) is $\text{tp}/(\text{fn} + \text{tp}) = 3/(28 + 3) = 0.097$. The specificity (1 minus False Positive Rate) is $\text{tn}/(\text{tn} + \text{fp}) = 59/(59 + 7) = 0.894$. Although the accuracy is good, the sensitivity (=power) of the SVM classifier is poor and the type II error (probability of accepting the null given it is false) is 0.903. SVM does not outperform the kNN classifier. Both classifiers give a similar confusion matrix. Neither of the 2 classifiers provides a sufficient sensitivity.

4.1 Unsupervised Learning

4.1.1 Hierarchical Clustering results

Euclidean distances between each of the 194 patients are calculated, and the average-linkage clustering has been performed on the first 4 principal component score vectors. The results were plotted as a dendrogram (not reproduced here). The dendrogram displays how patients are combined into clusters and is read from the bottom up. Each observation starts as its own cluster. Then the two observations that are closest are combined. This continues until all observations are combined into a single cluster. The dendrogram creates a hierarchical view of the similarity-dissimilarity among the 194 patients. Although nested clustering could be expected in the dataset, there is no obvious meaningful hierarchy in the dendrogram.

4.1.2 Partition Clustering results

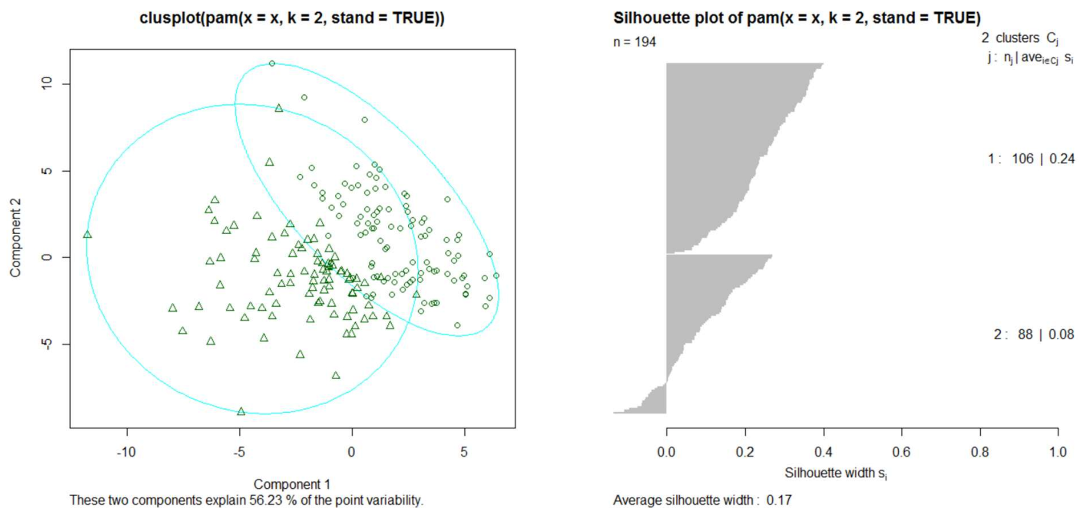

Figure 6: Bivariate plot for 2 principal components and partition around medoid clustering (left panel). Silhouette plot (right panel).

The cluster plot (left panel) for the two group partitioned around medoids clustering of the breast cancer data is displayed on Figure 6 along with the silhouette plot (right panel). The two medoids are actual observations contained in the dataset. In this case, they are observations 126 and 51 corresponding to patient ID 89812 and 862717 chosen to best represent the two clusters. The bivariate plot axes are the 2 first principal components derived from the 32 predictor variables. Each cluster is represented by an ellipse with the smallest area containing all its points. Note that observation 126 is of type R (= Recurrent) and observation 51 is of type N (= Non recurrent).

The agreement between the Tumor type (N or R) and the cluster is quantified using an adjusted Rand index (which provides a measure of the agreement between two partitions, adjusted for chance; it ranges from -1, no agreement, to 1, perfect agreement). The agreement between the observed tumor type (N or R) and the cluster solution is very poor : Rand index = 0.0248. The partitioned around medoids clustering appears not to be associated with the tumor type and is here meaningless in discriminating between R or N patients.

5 Discussion

Three predictors are selected by the Lasso shrinkage method in the prediction model for the Tumor Size. The Area Mean contribute positively in the prediction of the Tumor Size while the Symmetry Mean contribute negatively. The higher the Lymph Node number, the higher the Tumor Size. This is in line with our preliminary exploration illustrated on Figures 7 and 8 in the appendix.

The final CART tree is very simple, easy to interpret and is based on a single splitting rule using Lymph Node Status only.

Random Forest, less easy to interpret, did not improve the predictive power of the Tumor Size outcome variable.

The CART model precision is poorer than for the Lasso model and the Random Forest is poorer than the CART model (the standard deviation $\widehat{s.e.}$ for the tumor size estimation, on the test set, were 1.384, 1.862 and 2.192 centimeters respectively for Lasso, CART and Random Forest).

The supervised learning methods used for the classification of the Tumor Type (R or N) both show similar misclassification results on the test set : the misclassification errors are 33% with the k-Nearest Neighbours and 36% with the support vector machine with tuned radial kernel ($\gamma = 1$ and cost=1). The sensitivity (probability to detect a recurrent tumor when there is one) is very poor : 12.9% and 9.7% respectively, leading to the conclusion that the feature space (inputs) does not provide strong enough informative association to the targeted binary output.

The kNN method certainly suffers from the curse of dimensionality : spreading 194 observations over p=31 dimensions results in a phenomenon in which a given observation has in fact no nearby neighbors. The value $k = 3$ represents 1.55% of the volume of the observations to form a local average, but even with such a small volume fraction, 87% of the range of each input variable must be covered if the input data are uniformly distributed in the features space ($e_p(r) = 0.0155^{1/31} = 0.87$) as emphasized by Hastie et. al. ([5]-Chap.2.5, pp.22-23). So, the kNN classification method mainly yields a global (non-local) majority vote on the observed outcome directly, with poor sensitivity to the input local data. It is documented in the literature that non-parametric approaches like kNN often perform poorly when p is large ([4]-Chap 3.5, pp.108-109).

The poor results of SVM, as well, indicates that there appears to be no clear boundary of separation between the regions corresponding to each class. No improvement of the classification prediction rate were contributed by SVM on the dataset.

The analysis conducted here to find a good predictive model for the continuous output (tumor size) or a good classifier with high predictive power for the binary output (recurrent or non recurrent tumor) does not provide straightforward evidence that some strong predictor in the input space should be selected as highly informative or as significantly meaningful.

For the tumor size outcome prediction, the Lasso method selected 3 predictors out of the 31 from the feature space, namely : Area Mean, Symmetry Mean and Lymph Node Status. The Random Forest indicated that the 3 variables most contributing to variance reduction are Lymph Node Status, Symmetry SD and Symmetry Mean. In the regression tree, Lymph Node Status, Symmetry SD, Fractal Mean, Compactness SD, Smoothness Max and Concavity SD were among the top splitting variables used to build the initial tree but pruning left Lymph Node Status only.

The Lymph Node Status is the feature that is mostly associated to the Tumor Size. But as far as the other input variables are concerned, the data mining algorithms used did not provide a clear consistent overlap of the selected features among the 30 cytological measurements.

In the unsupervised learning approaches, without a ”teacher” to provide validation, the methods will always find clusters. One should be very cautious in the interpretation of the cluster analysis and self refrain to identify clusters that are ”real” in some sense rather than noticing a convenient partitioning.

The ability of cluster analysis to find erroneous clusters makes the validation step of cluster analysis important. Different clustering methods should be used, replicated with new samples. If the same clusters were consistently recovered, more confidence on the results would be gained.

In the breast cancer dataset cluster analysis, the partition clustering (PAM) was contrasted to the binary real observations and showed that there were no association (no agreement) between the Tumor Type observed classification and the clusters derived from the PAM algorithm with k=2.

Besides, the hierarchical clustering appears to be difficult to interpret with a large number of observations. The N and R binary outcome labels have been superimposed in the dendrogram showing that there is no obvious hierarchical pattern consistently discriminating the breast cancer type.

References

[1] Mangasarian, O.L and Wolberg, W. H. (2017). Available online on *http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wpbc*.

[2] Mangasarian, O.L and Wolberg, W. H. (1990). Cancer Diagnosis via Linear Programming. *SIAM News* **23**:1-18.

[3] Hastie, T et al. (2014). Glmnet Vignette available on line on *http://www.stanford.edu/h̃astie/glmnet/glmnet_alpha.html*.

[4] Gareth, J., Witten, D., Hastie, T., Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. Springer, New York.

[5] Hastie, T., Tibshirani, R., Friedman, J. (2009). *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*. Second Edition, Springer, New York.

[6] Kabacoff, R. (2015). *R in action. Data Analysis and graphics with R*. Second Edition, Manning, Shelter Island.

6 Appendix

6.1 Dataset full description

The approved dataset used in this project is called the Wisconsin Prognostic Breast Cancer dataset, available as text file on the UCI Machine Learning Repository [1]. Ten cytological characteristics on the cell nucleus collected in fine needle tissue aspiration are recorded. For each of the ten cytological characteristics, the mean, standard deviation and maximum values are recorded, providing a total of 30 continuous variables that are available as features space. The 10 cytological variables are :

1. Radius
2. Texture
3. Perimeter
4. Area
5. Smoothness (local variation in radius lengths)
6. Compactness (perimeter squared divided by area minus 1)
7. Concavity (severity of concave portions in the contour)
8. Concave points (number of concave portions in the contour)
9. Symetry
10. Fractal dimension

Each observation (patients) has 35 fields recorded. Field 1 is ID, Field 2 is the binary outcome (N or R), field 3 is the recurrence time or disease free time, fields 4-13 are the ten cytological variables (means), field 14-23 are the ten cytological standard deviations, field 24-33 are the ten cytological maximal values, field 34 is the tumor size in centimeters and field 35 is the lymph node status (number of positive axillary lymph nodes observed at time of surgery).
The 3 outcome variables are :

- Binary outcome : R = recurrent tumor or N = non-recurrent tumor : field 2.
- Continuous outcomes :
  1. Tumor size (diameter in cm) of the excised tumor (after surgery) : field 34.
  2. Recurrence time (in months) if binary outcome is R or disease free time if binary outcome is N : field 3.

6.2 Missing Data or Missing Attributes Exploration

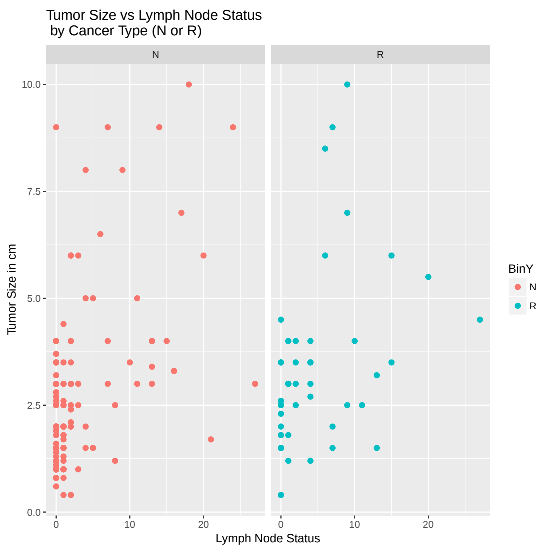

Figure 7: Lymph Node Status and Tumor Size faceted by Tumor Type (N in red or R in blue.)

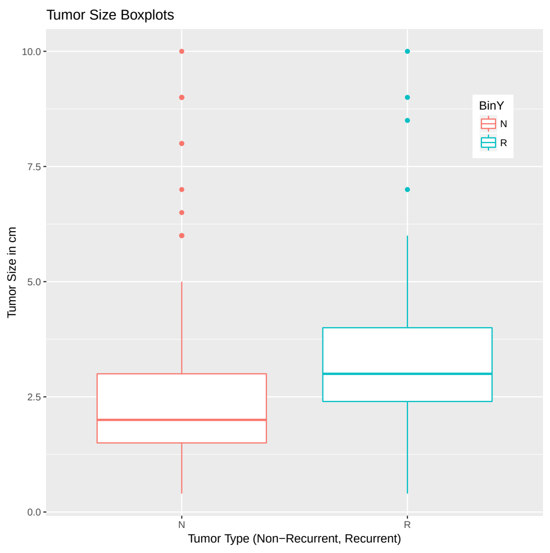

Figure 8: Association of Tumor Size at Time of Surgery to Tumor Type.

6.3 R scripts :

```
#-------------------------------------------------------------
# Continuous Outcome Prediction - First method
# SHRINKAGE Method - LASSO Regression
# Find a prediction model for Tumor Size versus Features space
#-------------------------------------------------------------
install.packages("glmnet")
library("glmnet")
# Keep the lymph node Status attribute but remove the 4
observations for which this attribute is missing
NoMissBC <- bc[is.na(bc$LymphNodeStatus)==FALSE,]
str(NoMissBC)
colrm <- c(-1,-2, -3) # remove ID, BinY and RecurTime
NoMissbrca <- NoMissBC[colrm]
names(NoMissbrca)
str(NoMissbrca)
x <- model.matrix(ContYTumorSize˜.,NoMissbrca)[, -31]
y <- NoMissbrca$ContYTumorSize
x
y
grid <- 10ˆseq(10, -2, length=100)
grid

set.seed(1)
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)
train
test
y.test <- y[test]
```

```
y.test
y.train <-y[train]
y.train

lasso.mod <- glmnet(x[train, ], y[train], alpha=1, lambda=grid)
plot(lasso.mod)

set.seed(1)
cv.out <- cv.glmnet(x[train, ], y[train], alpha=1)
pdf("Lasso.pdf")
plot(cv.out,
main="Training set Lambda tuning for Lasso\n L1 penalised regression on Tumor
Size outcome\n\n")
dev.off()

bestlambda <- cv.out$lambda.min
bestlambda
loglambda <- log(bestlambda)
loglambda
lasso.pred <- predict(lasso.mod, s=bestlambda, newx=x[test,])
lasso.pred
mean((lasso.pred-y.test)^2)
stddev <- sqrt(mean((lasso.pred-y.test)^2))
stddev

out <- glmnet(x, y, alpha=1, lambda=grid)
lasso.coef <- predict(out, type="coefficients", s=bestlambda)[1:32, ]
lasso.coef
lasso.coef[lasso.coef !=0]

# Compare observed and predicted values on the test set :
pdf("LassoPred.pdf")
plot(lasso.pred, y.test, xlim=c(0,11), ylim=c(0,11),
main="Comparison of LASSO estimated and observed\n Tumor Size in test set",
xlab="Y hat estimation for Tumor Size",
ylab="Y observed for Tumor Size")
abline(0,1)
dev.off()

#---------------------------------------------------------------
# Continuous Outcome Prediction - Second method
# REGRESSION TREE
# NON-PARAMETRIC METHOD
# Find a prediction model for Tumor Size versus Features space
#---------------------------------------------------------------
NoMissbrca
str(NoMissbrca)
names(NoMissbrca)

install.packages("tree")
library(tree)
set.seed(1234)
train <- sample(1:nrow(NoMissbrca), nrow(NoMissbrca)/2)
train
```

```
tree.brca <- tree(ContYTumorSize~., NoMissbrca, subset=train)
summary(tree.brca)
#pdf("InitialTree.pdf")
plot(tree.brca)
text(tree.brca, pretty=0)
#dev.off()

# prune trees and deviance :
pruned.brca.trees <- prune.tree(tree.brca)
pruned.brca.trees
summary(pruned.brca.trees)
plot(pruned.brca.trees)

# cross-validation :
cv.brca <- cv.tree(tree.brca, FUN=prune.tree)
cv.brca
summary(cv.brca)
pdf("TuningTree.pdf")
plot(cv.brca$size, cv.brca$dev, type="b",
main="Tuning for Optimal Size of the Tree",
xlab="Tree Size",
ylab="Deviance")
dev.off()

final.brca.tree <- prune.tree(tree.brca, best=2)

pdf("FinalTree.pdf")
plot(final.brca.tree)
text(final.brca.tree, pretty=0)
dev.off()
# predictions on the test set :
yhat <- predict(final.brca.tree, newdata=NoMissbrca[-train,])
brca.test <- NoMissbrca[-train, "ContYTumorSize"]
pdf("Compared.pdf")
plot(yhat, brca.test, xlim=c(0,11), ylim=c(0,11),
main="Comparison of estimated and observed\n Tumor Size in test set with 2
nodes tree",
xlab="Y hat estimation for Tumor Size",
ylab="Y observed for Tumor Size")
abline(0,1)
dev.off()

TestMSE <- mean((yhat-brca.test)^2)
TestMSE
seTestMSE <- sqrt(TestMSE)
seTestMSE

#------------------------------------------------------------
# Continuous Outcome Prediction - Third method
# RANDOM FOREST
# NON-PARAMETRIC METHOD
# Find a prediction model for Tumor Size versus Features space
#------------------------------------------------------------
library(randomForest)
```

```
train
length(train)
# RANDOM FOREST :
set.seed(1234)
rf.brca <- randomForest(ContYTumorSize~., data=NoMissbrca,
subset=train, mtry=6, ntree=2000, importance=TRUE)
rf.brca
yhat.rf <- predict(rf.brca, newdata=NoMissbrca[-train,])
pdf("rf.pdf")
plot(yhat.rf, brca.test, xlim=c(0,12), ylim=c(0,12),
main="Comparison of Random Forest estimated and observed\n
Tumor Size in test set",
xlab="Y hat estimation for Tumor Size",
ylab="Y observed for Tumor Size")
abline(0,1)
dev.off()
TestMSE <- mean((yhat.rf-brca.test)^2)
TestMSE
seTestMSE <- sqrt(TestMSE)
seTestMSE

importance(rf.brca, type=1)
pdf("VarImp.pdf")
varImpPlot(rf.brca, type=1)
dev.off()

#-------------------------------------------------------------
# Classifcation
#-------------------------------------------------------------
library(class)
#------------ prepare data and standardize the data :
# Keep the lymph node Status attribute but remove the 4
observations for which this attribute is missing
NoMissBC <- bc[is.na(bc$LymphNodeStatus)==FALSE,]
str(NoMissBC)
colrm <- c(-1, -2, -3)
x <- NoMissBC[colrm]
#-------------------------------------------------------------
# Standardization (scaling) :
#-------------------------------------------------------------
standardized.x <- scale(NoMissBC[colrm])
standardized.x
var(x[, 1])
var(standardized.x[, 1])
#-------------------------------------------------------------
# Training set and Test set split :
#-------------------------------------------------------------
set.seed(7007)
train <- sample(194, 97)
x.train <- standardized.x[train,]
x.test <- standardized.x[-train,]
str(x.train)
y.train <- NoMissBC$BinY[train]
y.train
```

```
y.test <- NoMissBC$BinY[-train]
y.test
#--------------------------------------------------------------
# Binary Outcome Prediction - First Classification method
# k Nearest Neighbors
#--------------------------------------------------------------
#--- KNN :
set.seed(7)
#--- KNN with K=1 :
knn.pred <- knn(x.train, x.test, y.train, k=1)
knn.pred
t <- table(knn.pred, y.test)
mean(y.test==knn.pred)
tn <- t[1, 1]
fn <- t[1, 2]
fp <- t[2, 1]
tp <- t[2, 2]
sensitivity1 <- tp/(tp + fn)
specificity1 <- tn/(tn + fp)
accuracy1 <- (tp + tn)/(tp + tn + fp +fn)
accuracy1

#--- KNN with K=2 :
knn.pred <- knn(x.train, x.test, y.train, k=2)
table(knn.pred, y.test)
mean(y.test==knn.pred)
t <- table(knn.pred, y.test)
tn <- t[1, 1]
fn <- t[1, 2]
fp <- t[2, 1]
tp <- t[2, 2]
sensitivity2 <- tp/(tp + fn)
specificity2 <- tn/(tn + fp)
accuracy2 <- (tp + tn)/(tp + tn + fp +fn)
accuracy2

#--- KNN with K=3 :
knn.pred <- knn(x.train, x.test, y.train, k=3)
table(knn.pred, y.test)
mean(y.test==knn.pred)
t <- table(knn.pred, y.test)
tn <- t[1, 1]
fn <- t[1, 2]
fp <- t[2, 1]
tp <- t[2, 2]
sensitivity3 <- tp/(tp + fn)
specificity3 <- tn/(tn + fp)
accuracy3 <- (tp + tn)/(tp + tn + fp +fn)
accuracy3
sensitivity3
specificity3

errate <- numeric(7)
flexibility <- numeric(7)
```

```
sensitivity <- numeric(7)
specificity <- numeric(7)
accuracy <- numeric(7)

for (i in 1:7){
flexibility[i] <- 1/i
knn.pred <- knn(x.train, x.test, y.train, k=i)
t <- table(knn.pred, y.test)
errate[i] <- 1- mean(y.test==knn.pred)
tn <- t[1, 1]
fn <- t[1, 2]
fp <- t[2, 1]
tp <- t[2, 2]
sensitivity[i] <- tp/(tp + fn)
specificity[i] <- tn/(tn + fp)
accuracy[i] <- (tp + tn)/(tp + tn + fp +fn)
}

yscale = c(0, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40)
pdf("kNN.pdf")
par(mar=c(5, 4, 4, 8) +0.1)
plot(flexibility, errate, type="b", pch=21, col="black", yaxt="n",
ann=FALSE, ylim=c(0, 0.40))
lines(flexibility, sensitivity, pch=21, col="red", yaxt="n", lty=2, type="b")
axis(2, at=yscale, labels=round(yscale, digits=2), col.axis="black", las=2)
axis(4, at=yscale, labels=round(yscale, digits=2), col.axis="red", las=2)
mtext("Sensitivity", side=4, line=3, cex.lab=1, las=0, col="red")
title("k Nearest Neighbor Test Error Rate\n and Sensitivity vs Flexibility",
xlab="Flexibility = 1/k", ylab="Test Error Rate")
par(opar)
dev.off()

sensitivity
specificity
accuracy
errate
#------------------------------------------------------------
# Binary Outcome Prediction - Second Classification method
# SVM with radial kernel
#------------------------------------------------------------
install.packages("e1071")
library(e1071)

# The data have been standardized before

dat <- data.frame(x=x.train, y=as.factor(y.train))
str(dat)
testdat <- data.frame(x=x.test, y=as.factor(y.test))
str(testdat)

# SVM : radial kernel :

out <- svm(y˜., data=dat, kernel="radial", gamma=1, cost=1, scale=TRUE)
summary(out)
```

```
table(out$fitted, dat$y)
accuracy <- (82+15)/(97)
accuracy

# decrease the cost :
svmfit <- svm(y˜., data=dat, kernel="radial", gamma=0.5, cost=0.5, scale=TRUE)
summary(svmfit)
table(svmfit$fitted, dat$y)

accuracy <- (82)/(97)
accuracy


# tuning cost :
set.seed(1)
tune.out <- tune(svm, y˜., data=dat, kernel="radial", scale=TRUE,
ranges=list(cost=c(1, 10, 100, 1000),
gamma=c(0.1, 0.5, 1, 1.5, 2, 3, 5)))
summary(tune.out)

bestmod <- tune.out$best.model
bestmod
summary(bestmod)

str(testdat)
head(testdat)

#--- Misclassification error :
table(true=testdat$y, pred=predict(tune.out$best.model, newx=testdat))

table(testdat$y)
tot <- 59+3+7+28
mis<-28+7
mis/tot

#---------------------------------------------------------------
# UNSUPERVISED LEARNING
# Hierarchical clustering :
#---------------------------------------------------------------
install.packages("hclust")
library("hclust")
hc.complete <- hclust(dist(x), method="complete")
hc.single <- hclust(dist(x), method="single")
hc.average <- hclust(dist(x), method="average")
hc.complete
hc.single
hc.average

plot(hc.complete, cex=.9)
plot(hc.single, cex=.9)
plot(hc.average, cex=.9)
cutree(hc.average, 4)
opar
xsc=scale(x)
```

```
hc.average <- hclust(dist(xsc), method="average")

plot(hclust(dist(xsc), method="average"), hang=-1, cex=.8,
main="Hierarchical Average Linkage Clustering with scaled
features")

ct.hclust <-table(Type, cutree(hc.average,4))
ct.hclust
install.packages("flexclust")
library(flexclust)
randIndex(ct.hclust)

install.packages("NbClust")
library(NbClust)
devAskNewPage(ask=TRUE)
nc <- NbClust(xsc, distance="euclidean", min.nc=2, max.nc=15,
method="average")
table(nc$Best.n[1,])
clusters <- cutree(hc.average, k=9)
table(clusters)
aggregate(x, by=list(cluster=clusters), median)
plot(hc.average, hang=-1, cex=.8, main="Average Linkage
Clustering\n 9 Cluster Solution")
rect.hclust(hc.average, k=9)


# perform hierarchical clustering on the 4 PC score vectors :
pr.out <- prcomp(x, scale=TRUE)
pr.out
summary(pr.out)
plot(pr.out)
hc.out <- hclust(dist(pr.out$x[, 1:4]))
Type <- toupper(Type)
Type
pdf("hierarchyPC4.pdf")
plot(hc.out, labels=Type, hang=0, main="Hier. Clust. on First
4 Score Vectors")
dev.off()

#-----------------------------------------------------------------
# UNSUPERVISED LEARNING
# Partition Clustering with PAM (Partition around Medoids):
#-----------------------------------------------------------------
head(x)
str(x)
Type <- factor(NoMissBC$BinY)
Type
BinX <- cbind(Type, x)
head(BinX)
library(cluster)
set.seed(1234)
fit.pam <- pam (x, k=2, stand=TRUE)
fit.pam$medoids
NoMissBC$ID[126]
```

```
NoMissBC$ID[51]
opar
par(mfrow=c(1, 2))
clusplot(fit.pam, main="Bivariate Cluster Plot")
ct.pam <-table(Type, fit.pam$clustering)
ct.pam
install.packages("flexclust")
library(flexclust)
randIndex(ct.pam)
par(mfrow=c(1, 2))
plot(fit.pam)
opar
Type[126]
Type[51]
```