Breast Cancer Wisconsin (Diagnostic) Dataset to Predict Cancer Diagnosis Based on Cell Features

Atena Rashidi

Content

I. Introduction

1. Project Description

Cancer occurs as a result of mutations, or abnormal changes, in the genes responsible for regulating the growth of cells and keeping them healthy. Breast cancer is an uncontrolled growth of breast cells. A tumor can be benign (not dangerous to health) or malignant (has the potential to be dangerous). The term "breast cancer" refers to a malignant tumor that has developed from cells in the breast. This project 's aim is to detect breast cancer by predicting whether a tumor is benign or malignant.

2. Dataset

In this project Breast Cancer Wisconsin (Diagnostic) Data Set [1] is used to predict whether the cancer is benign or malignant. This dataset has been created by Dr. William H. Wolberg from General Surgery Department, W. Nick Street and Olvi L. Mangasarian from Computer Sciences Department in University of Wisconsin [2]. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

id	diagnosis	radius_mean	…	radius_se	area_worst	concavity_worst
842302	M	17.99	…	1.095	2019	0.7119
842517	M	20.57	…	0.5435	1956	0.2416
84300903	M	19.69	…	0.7456	1709	0.4504
84358402	M	20.29	…	0.7572	1575	0.4
843786	M	12.45	…	0.3345	741.6	0.5355
…	…	…	…	…	…	…

Attribute Information:

1) ID number
2) Diagnosis (M = malignant, B = benign)
3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness ($perimeter^2$ / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry

j) fractal dimension ("coastline approximation" - 1)
These features are selected for model fitting: CT, UCSize, UCShape, MA, SECS, BC, NN, and M.

II. Methodology

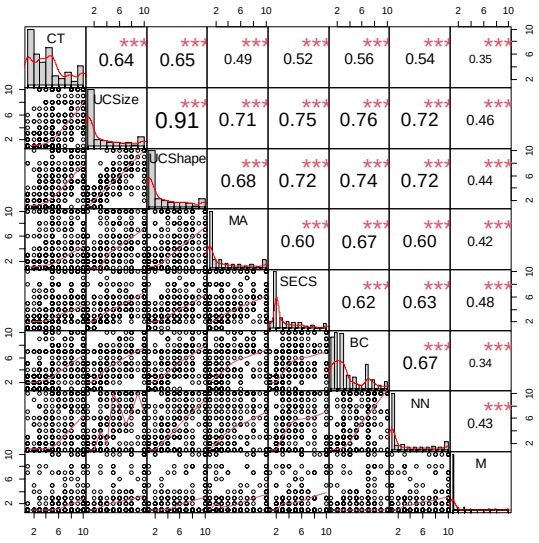
To predict whether the cancer is Benign or Malignant a classification problem should be solved. Several models such as: Generalize Linear Model/Logistic Regression, Gradient Boosting Classifier, Random Forest Classifier, Decision Tree Classifier, K Neighbors Classifier, Support Vector Machine classifier, and MLP Classifier

1. Data Preparation

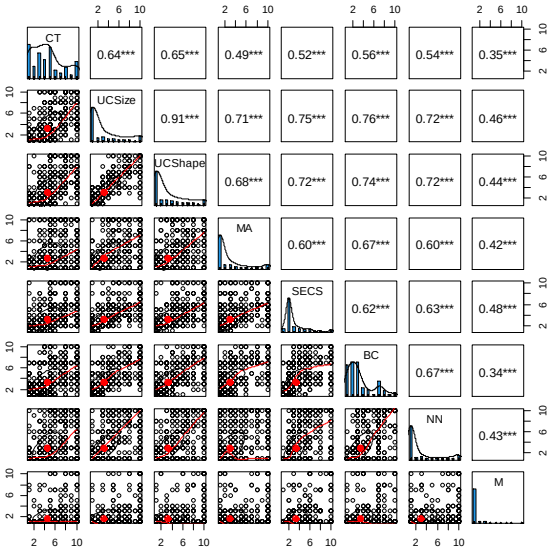
First, necessary libraries will be loaded and then the dataset will be read from the URL Breast Cancer data from Wisconsin. Then the dataset will be cleaned of the id column and column with "?" values. Then the data type will be checked. All the data type is integer but diagnosis. So, the target, which is diagnosis of Benign or Malignant should change to a dummy variable, named outcome.

1.1. Correlation Chart for Means and Standard Error

In this part, the correlation for the means and the standard error are plotted.



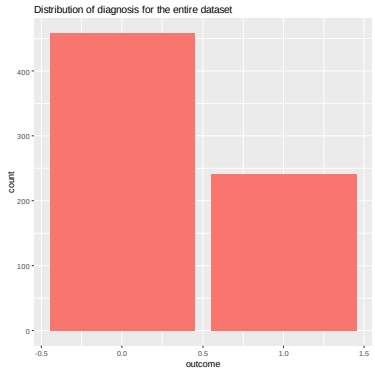
Correlation for mean

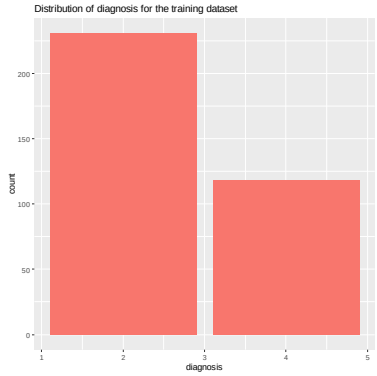


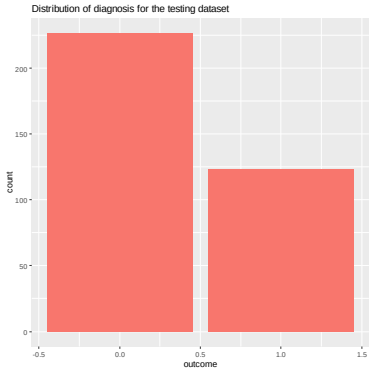
Correlation for standard error

1.2.Splitting the dataset into the training sample and the testing sample

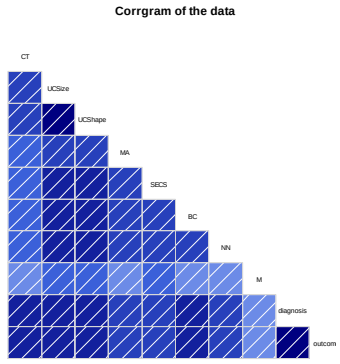
To evaluate the performance of the model, the dataset is split into the training sample (%50) and the testing sample (%50). In what follows, the distribution of diagnosis for the entire dataset, training, and testing samples are shown.

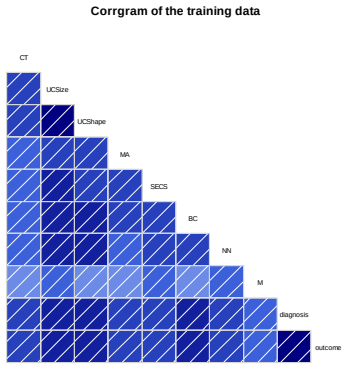


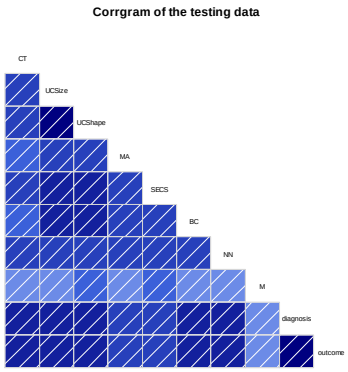




The corrgram of the dataset, training, and testing samples are shown to produce a graphical display of a correlation matrix.

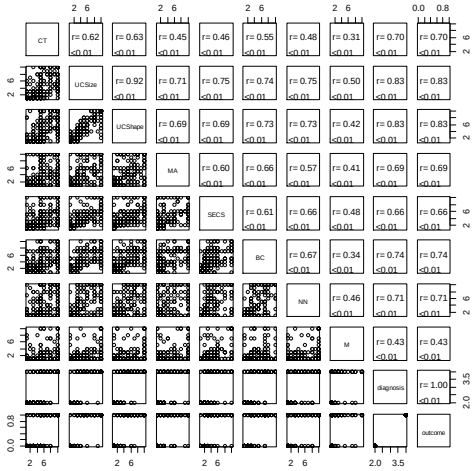






1.3.Coefficients and p-values

In this plot the correlation along with the p-value for less than 0.01 is shown:



2. Breast Cancer Data with Fitting Several Models

In this part, several models such as: Generalize Linear Model/Logistic Regression, Gradient Boosting Classifier, Random Forest Classifier, Decision Tree Classifier, K Neighbors Classifier, Support Vector Machine classifier, and MLP Classifier.

2.1.Fitting a model using Generalize Linear Model/Logistic Regression function

1. Using all features as predictors of diagnostic

Coefficients:

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-10.1886	1.56226	-6.522	6.95e-11
CT	0.58717	0.16233	3.617	0.000298
UCSize	0.06917	0.30365	0.228	0.819800
UCShape	0.61346	0.28887	2.124	0.033700
MA	0.16958	0.14388	1.179	0.238545
SECS	0.29788	0.25076	1.188	0.234871
BC	0.52313	0.22373	2.338	0.019375
NN	0.07228	0.15185	0.476	0.634084
M	0.64951	0.34785	1.867	0.061872

More information is in Appendix.

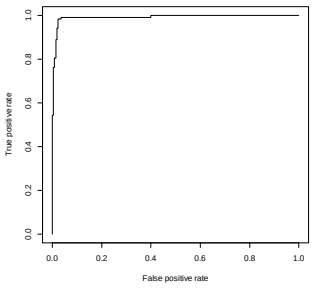
2. Using Uniform Cell size and Uniform Cell Shape as predictors of diagnosis

Coefficients:

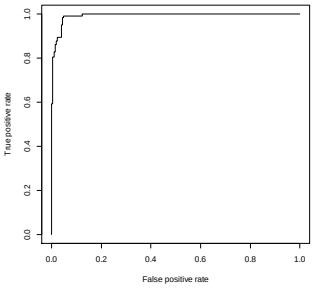
	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-5.6790	0.6091	-9.323	< 2e-16
UCSize	0.8729	0.2412	3.619	0.000296
UCShape	0.8573	0.2238	3.831	0.000127

The area under the ROC curve is called as AUC -Area Under Curve. AUC ranges between 0 and 1 and is used for successful classification of the logistics model.

	Training Sample	**Testing Sample**	**AUC for Testing Samples**	**AUC for Testing Samples**
Model Accuracy	0.957	0.934	0.983	0.934



train



test

The logistic regression confusion matrix and test error are as follows:

y.test / logit.pred	0: Real-Benign	1: Real-Malignant
Predicted-Benign	218	7
Predicted-Malignant	9	116
logit.test.error	0.04571429	

2.2.Fitting a model using LDA and QDA

LDA and QDA algorithms are based on Bayes theorem and are different in their approach for classification from the Logistic Regression.

LDA-Confusion Matrix

y.test / logit.pred	0: Real-Benign	1: Real-Malignant
Predicted-Benign	222	13
Predicted-Malignant	5	110
lda.test.error	0.05142857	

K-fold=5

Mean of lda.test.error	0.04722508

QDA-Confusion Matrix

y.test / logit.pred	0: Real-Benign	1: Real-Malignant
Predicted-Benign	216	3
Predicted-Malignant	11	120
qda.test.error	0.04	

K-fold=5

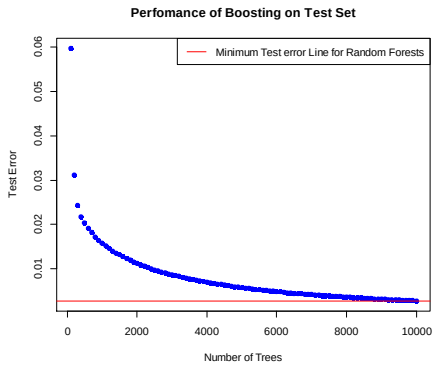
Mean of qda.test.error	0.05434738

2.3. Fitting a model using Gradient Boosting

The gradient boosting is done for 10000 trees in every 100 trees. The results are as follows:

Mean squared test error for each of the 100 trees averaged:

100	200	300	400	500	600
0.05962488	0.03113548	0.02437623	0.02175681	0.02042129	0.01921702



2.4.Fitting a model using KNN

In this part KNN method is applied for k = 5, 20, 50, and 100.

K=5 - Confusion Matrix

y.test / logit.pred	0: Real-Benign	1: Real-Malignant
Predicted-Benign	219	6
Predicted-Malignant	8	117
K5.test.error	0.04	

K=20 - Confusion Matrix

y.test	0: Real-Benign	1: Real-Malignant

logit.pred		
Predicted-Benign	219	8
Predicted-Malignant	8	115
K20.test.error	0.04571429	

K=50 - Confusion Matrix

y.test / logit.pred	0: Real-Benign	1: Real-Malignant
Predicted-Benign	220	13
Predicted-Malignant	7	110
K50.test.error	0.05714286	

K=100 - Confusion Matrix

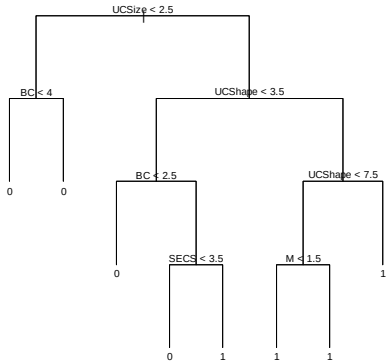
y.test / logit.pred	0: Real-Benign	1: Real-Malignant
Predicted-Benign	223	26
Predicted-Malignant	4	97
K100.test.error	0.08571429	

2.5.Fitting a model using Decision Tree

In this step, a decision tree model is fitted and because M and B should be classified, it is a classification tree problem.

Classification tree:

tree(formula = outcome ~ UCSize + UCShape + MA + SECS + BC + NN + M, data = training)
Variables actually used in tree construction:
[1] "UCSize" "BC" "UCShape" "SECS" "M"
Number of terminal nodes: 8
Residual mean deviance: 0.2197 = 74.93 / 341
Misclassification error rate: 0.04011 = 14 / 349

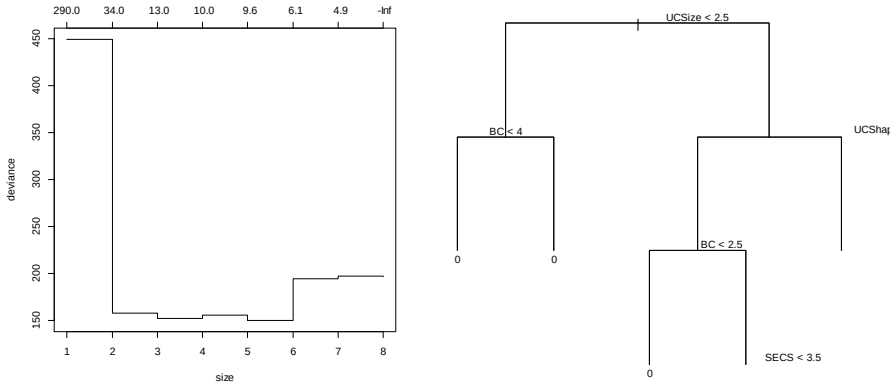


In this steep, the tree is prune to avoid over fitting. Then the probability for each class will be calculated by prediction. After that the tree will be pruned to avoid over fitting.

Classification tree:

snip.tree(tree = model_tree, nodes = c(7L, 13L))
Variables actually used in tree construction:
[1] "UCSize" "BC" "UCShape"
Number of terminal nodes: 5
Residual mean deviance: 0.2777 = 95.53 / 344
Misclassification error rate: 0.04585 = 16 / 349

Seems like a tree of size 3 might be best.

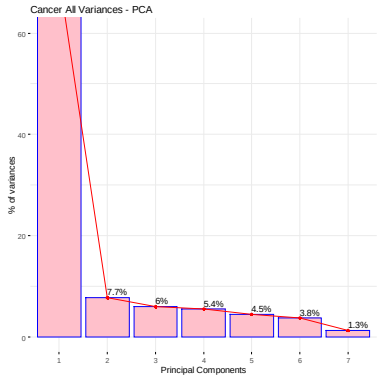


To show the importance of the components, PCA is calculated.

Importance of components: CT, UCSize, UCShape

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Standard deviation	**2.2975**	**0.86438**	**0.73405**	0.63905	0.61290	0.54618	0.51251	0.3007
Proportion of Variance	0.6598	0.09339	0.06735	0.05105	0.04696	0.03729	0.03283	0.0113
Cumulative Proportion	0.6598	0.75322	0.82057	0.87162	0.91858	0.95586	0.98870	1.0000

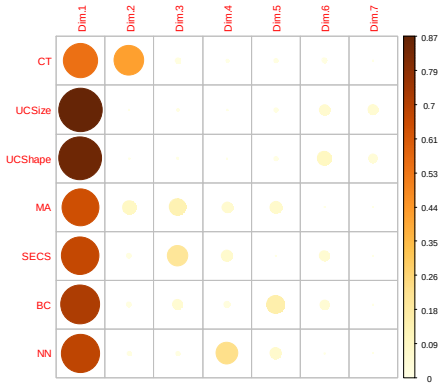
The percentage of variance for the principle component is shown as follows:



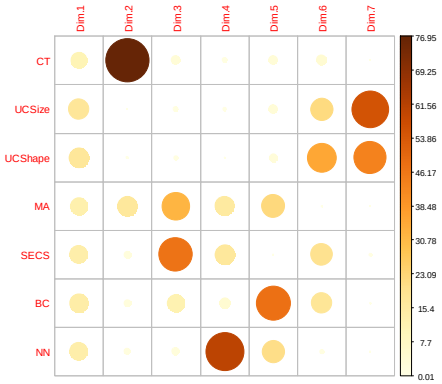
Principal component analysis results for variables:

The components of the get_pca_var() can be used in the plot of variables as follow:

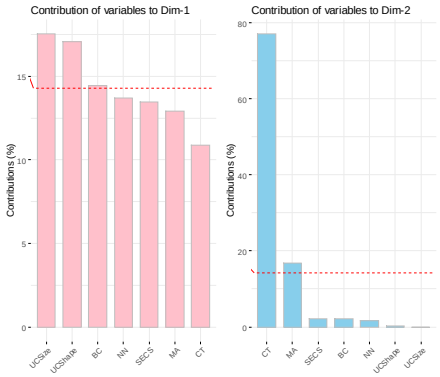
- var$coord: coordinates of variables to create a scatter plot
- var$cos2: represents the quality of representation for variables on the factor map. It's calculated as the squared coordinates: var.cos2 = var.coord * var.coord.
- var$contrib: contains the contributions (in percentage) of the variables to the principal components. The contribution of a variable (var) to a given principal component is (in percentage): (var.cos2 * 100) / (total cos2 of the component).

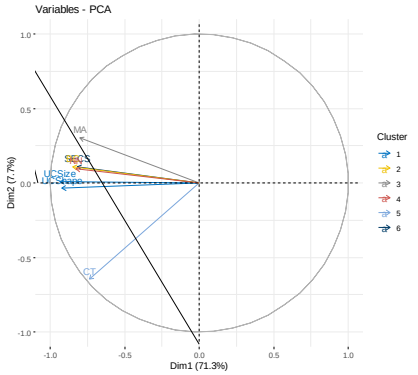


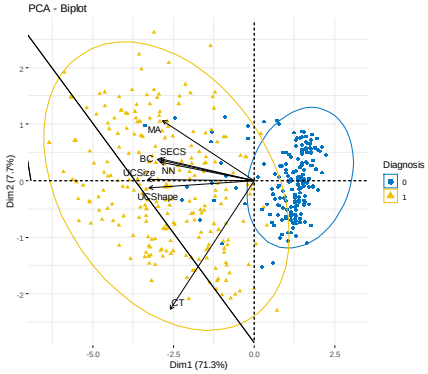
var$cos2



var$contrib







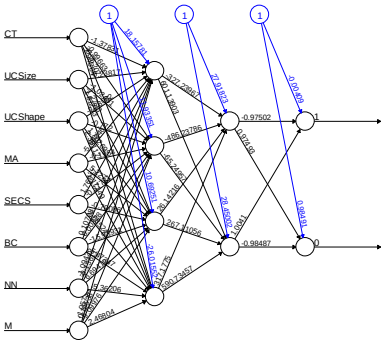
2.6. Fitting a model using MLP

unit definition section :

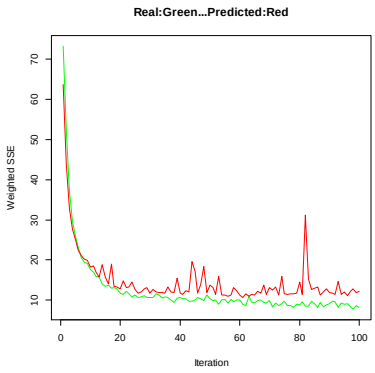
no.	typeName	unitName	act	bias	st	position	act func	out func	sites
1		Input_CT	4.00000	0.02471	i	1,0,0	Act_Identity		
2		Input_UCSize	8.00000	-0.17532	i	2,0,0	Act_Identity		
3		Input_UCShape	8.00000	-0.14820	i	3,0,0	Act_Identity		
4		Input_MA	5.00000	-0.08374	i	4,0,0	Act_Identity		
5		Input_SECS	4.00000	-0.04494	i	5,0,0	Act_Identity		
6		Input_BC	10.00000	-0.17657	i	6,0,0	Act_Identity		
7		Input_NN	4.00000	0.07603	i	7,0,0	Act_Identity		
8		Input_M	1.00000	-0.07401	i	8,0,0	Act_Identity		
9		Hidden_2_1	0.99638	-2.52293	h	1,2,0	|||		
10		Hidden_2_2	0.98535	-1.12899	h	2,2,0	|||		
11		Hidden_2_3	0.99439	-0.76642	h	3,2,0	|||		
12		Hidden_2_4	0.06883	4.26625	h	4,2,0	|||		
13		Hidden_2_5	0.00225	3.33527	h	5,2,0	|||		
14		Output_1	0.95294	-1.76385	o	1,4,0	|||		

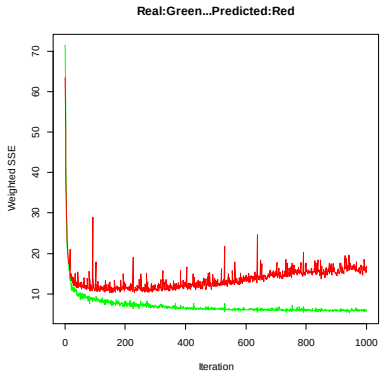
connection definition section :

target	site	source:weight
9		8: 0.11286, 7:-0.06200, 6: 0.17608, 5:-0.01334, 4:-0.01733, 3: 0.43180, 2: 0.30544, 1: 0.18937
10		8: 0.05240, 7:-0.05504, 6: 0.06018, 5: 0.01493, 4:-0.01099, 3: 0.25871, 2: 0.28547, 1: 0.13627
11		8: 0.10540, 7:-0.08153, 6: 0.14434, 5:-0.09727, 4:-0.07996, 3: 0.36298, 2: 0.27489, 1: 0.10179
12		8:-0.29477, 7:-0.15382, 6:-0.11435, 5:-0.21848, 4:-0.41021, 3:-0.13644, 2: 0.00658, 1:-0.21341
13		8:-0.14225, 7: 0.02221, 6:-0.17749, 5: 0.01041, 4:-0.01181, 3:-0.40425, 2:-0.43337, 1:-0.22030
14		13:-3.65317, 12:-4.41662, 11: 1.25607, 10: 1.42037, 9: 2.44448



Error: 2.925122 Steps: 11963





Overfitting

2.7.Fitting a model using Random Forest

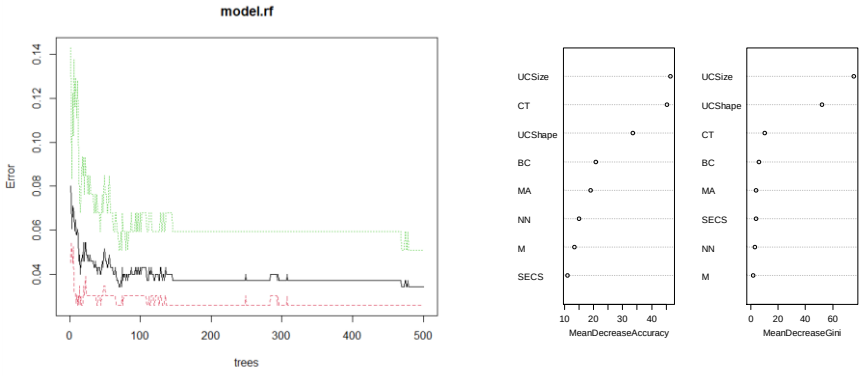
Type of random forest: classification

Number of trees: 500

OOB estimate of error rate: 3.72%

Confusion Matrix

y.test / logit.pred	0: Real-Benign	1: Real-Malignant
Predicted-Benign	223	8
Predicted-Malignant	4	115
rf.class.error → 0	0.02597403	
rf.class.error → 1	0.05084746	
error	0.0429	



The important features are in order as follows:

importance(rf.model)

	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
CT	16.779876	15.000901	19.76880	18.369091
UCSize	11.682101	15.790080	18.81817	20.673774
UCShape	12.988556	18.965458	20.81360	27.861230
MA	10.668946	7.143136	12.58421	17.420100
SECS	8.756312	8.456877	11.98538	15.384029
BC	10.326956	12.766176	14.23018	20.190614
NN	10.538682	6.946311	11.96931	15.987237
M	7.519154	7.595463	9.76295	8.883679

2.8.Fitting a model using SVM

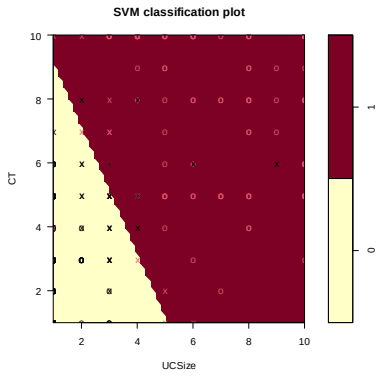
Based on the principle component analysis (PCA), and important features in ransom forest the important components are CT and UCSize respectively. Therefore, the support vector classifier and support vector machine are fitted to the dataset based on these two features.

A. **Fit Support Vector Classifier using a linear kernel**

When the cost is 10, the prediction is better. A 10-fold CV is used to find the best cost:

Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters: **cost = 10**
- best performance: 0.04882353
- Detailed performance results:

	cost	error	dispersion
1	1e-03	0.33781513	0.08510982
2	1e-02	0.08042017	0.03335581
3	1e-01	0.06025210	0.02143562
4	1e+00	0.05739496	0.02368248
5	5e+00	0.05168067	0.02984278
6	**1e+01**	**0.04882353**	**0.03061933**
7	1e+02	0.04882353	0.03061933

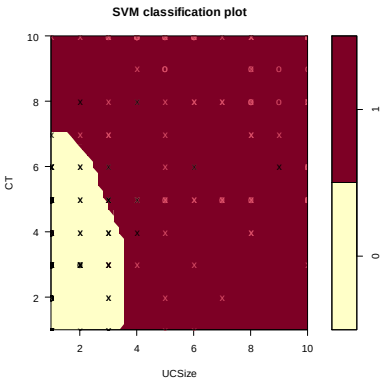


B. Fit Support Vector Classifier using a linear radial

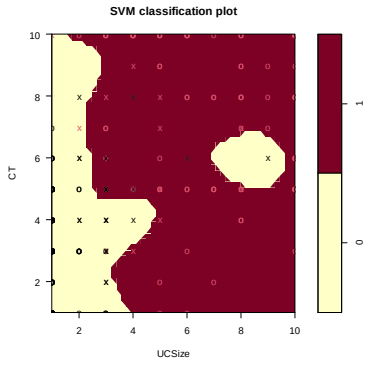
When the cost is 0.1, the prediction is better. A 10-fold CV is used to find the best cost:

Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters: **cost = 0.1 gamma = 4**
- best performance: 0.04008403
- Detailed performance results:

	cost	gamma	error	dispersion
1	1e-01	0.5	0.04882353	0.02397357
2	1e+00	0.5	0.04882353	0.03061933
3	1e+01	0.5	0.04882353	0.03345070
4	1e+02	0.5	0.05453782	0.03447188
5	1e+03	0.5	0.06025210	0.03934363
6	1e-01	1.0	0.04882353	0.03061933
7	1e+00	1.0	0.05453782	0.03447188
8	1e+01	1.0	0.05453782	0.03447188
9	1e+02	1.0	0.06016807	0.04353053
10	1e+03	1.0	0.06016807	0.04556658
11	1e-01	2.0	0.04596639	0.03108625
12	1e+00	2.0	0.05168067	0.03274137
13	1e+01	2.0	0.05731092	0.03809895
14	1e+02	2.0	0.06016807	0.04353053
15	1e+03	2.0	0.06016807	0.04353053
16	**1e-01**	**3.0**	**0.05159664**	**0.03515646**
17	1e+00	3.0	0.05453782	0.03447188
18	1e+01	3.0	0.05445378	0.03916924
19	1e+02	3.0	0.06016807	0.04353053
20	1e+03	3.0	0.06016807	0.04353053
21	1e-01	4.0	0.04008403	0.03067975
22	1e+00	4.0	0.04873950	0.03318116
23	1e+01	4.0	0.05445378	0.03422598
24	1e+02	4.0	0.05445378	0.03422598
25	1e+03	4.0	0.05445378	0.03422598



cost = 0.1 gamma = 4



cost = 1000 gamma = 1

III. Conclusion

To find out the best methodology, the accuracy of different approaches after data preparation will is calculated. Among all these methods, first gradient boosting and then SVM → linear had the lowest error.

IV. References

[1] https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/

[2] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

[4] Breast Cancer Analysys in R (rstudio-pubs-static.s3.amazonaws.com)

[5] https://rdrr.io/cran/RSNNS/man/mlp.html

[6] mlp: Create and train a multi-layer perceptron (MLP) in RSNNS: Neural Networks using the Stuttgart Neural Network Simulator (SNNS) (rdrr.io)

[7] https://rdrr.io/cran/RSNNS/src/R/mlp.R

[8] https://www.geeksforgeeks.org/random-forest-approach-for-classification-in-r-programming/?ref=rp

[9] https://www.geeksforgeeks.org/how-neural-networks-are-used-for-classification-in-r-programming/?ref=rp

[10] http://127.0.0.1:10039/library/RSNNS/html/mlp.html