Breast Cancer Prediction

Wedad

2023-11-02

## Loading the Libraries

library(mlbench)  
library(caTools)  
library(rpart)  
library(rpart.plot)  
library(plotly)  
library(e1071)  
library(ggplot2)  
library(caret)  
library(pROC)  
library(PRROC)  
library(xgboost)

## Loading the Dataset

# loading the dataset  
data("BreastCancer")  
  
# checking the structure of the dataset  
str(BreastCancer)

## 'data.frame': 699 obs. of 11 variables:  
## $ Id : chr "1000025" "1002945" "1015425" "1016277" ...  
## $ Cl.thickness : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 5 5 3 6 4 8 1 2 2 4 ...  
## $ Cell.size : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 1 1 2 ...  
## $ Cell.shape : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 2 1 1 ...  
## $ Marg.adhesion : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 5 1 1 3 8 1 1 1 1 ...  
## $ Epith.c.size : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 2 7 2 3 2 7 2 2 2 2 ...  
## $ Bare.nuclei : Factor w/ 10 levels "1","2","3","4",..: 1 10 2 4 1 10 10 1 1 1 ...  
## $ Bl.cromatin : Factor w/ 10 levels "1","2","3","4",..: 3 3 3 3 3 9 3 3 1 2 ...  
## $ Normal.nucleoli: Factor w/ 10 levels "1","2","3","4",..: 1 2 1 7 1 7 1 1 1 1 ...  
## $ Mitoses : Factor w/ 9 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 5 1 ...  
## $ Class : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...

# View the entire dataset   
View(BreastCancer)

The data has 699 obs. of 11 variables, The objective is to identify each of a number of benign or malignant classes. Samples arrive periodically as Dr. Wolberg reports his clinical cases. The database therefore reflects this chronological grouping of the data. This grouping information appears immediately below, having been removed from the data itself. Each variable except for the first was converted into 11 primitive numerical attributes with values ranging from 0 through 10. There are 16 missing attribute values. A data frame with 699 observations on 11 variables, one being a character variable, 9 being ordered or nominal, and 1 target class.

[,1] Id Sample code number #[,2] Cl.thickness Clump Thickness #[,3] Cell.size Uniformity of Cell Size #[,4] Cell.shape Uniformity of Cell Shape #[,5] Marg.adhesion Marginal Adhesion #[,6] Epith.c.size Single Epithelial Cell Size #[,7] Bare.nuclei Bare Nuclei #[,8] Bl.cromatin Bland Chromatin #[,9] Normal.nucleoli Normal Nucleoli #[,10] Mitoses Mitoses #[,11] Class Class

#remove the first column,   
BreastCancer<-BreastCancer[,-1]  
  
# show the summary of the dataset  
summary(BreastCancer)

## Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size  
## 1 :145 1 :384 1 :353 1 :407 2 :386   
## 5 :130 10 : 67 2 : 59 2 : 58 3 : 72   
## 3 :108 3 : 52 10 : 58 3 : 58 4 : 48   
## 4 : 80 2 : 45 3 : 56 10 : 55 1 : 47   
## 10 : 69 4 : 40 4 : 44 4 : 33 6 : 41   
## 2 : 50 5 : 30 5 : 34 8 : 25 5 : 39   
## (Other):117 (Other): 81 (Other): 95 (Other): 63 (Other): 66   
## Bare.nuclei Bl.cromatin Normal.nucleoli Mitoses Class   
## 1 :402 2 :166 1 :443 1 :579 benign :458   
## 10 :132 3 :165 10 : 61 2 : 35 malignant:241   
## 2 : 30 1 :152 3 : 44 3 : 33   
## 5 : 30 7 : 73 2 : 36 10 : 14   
## 3 : 28 4 : 40 8 : 24 4 : 12   
## (Other): 61 5 : 34 6 : 22 7 : 9   
## NA's : 16 (Other): 69 (Other): 69 (Other): 17

## Preprocessing the Dataset

### Checking the Null Values

Sometimes R does not recognize empty strings and question marks as null values, so we first replace then with nulls if any then remove all the nulls.

# Replace empty strings with NA  
BreastCancer[BreastCancer == ""] <- NA  
  
# Replace ? with NA  
BreastCancer[BreastCancer == "?"] <- NA  
  
# Check for null values in the BreastCancer dataset  
null\_values <- sum(is.null(BreastCancer$Bare.nuclei))  
  
print(paste("Number of null values in the BreastCancer dataset:", null\_values))

## [1] "Number of null values in the BreastCancer dataset: 0"

# remove nulls  
BreastCancer <- na.omit(BreastCancer)

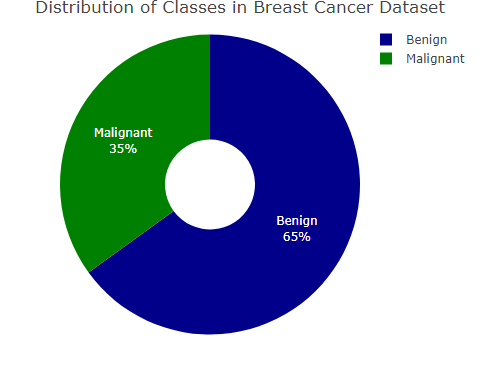
Seems we have no null values. Having confirmed that, we can now proceed with the analysis

### Encoding the Class Variable

The next step is to encode the class variable to 0, and 1.

# # Encode Class variable as 0 and 1  
# BreastCancer$Class <- ifelse(BreastCancer$Class == "benign", 0, 1)  
#   
# # Verify the changes  
# unique(BreastCancer$Class)

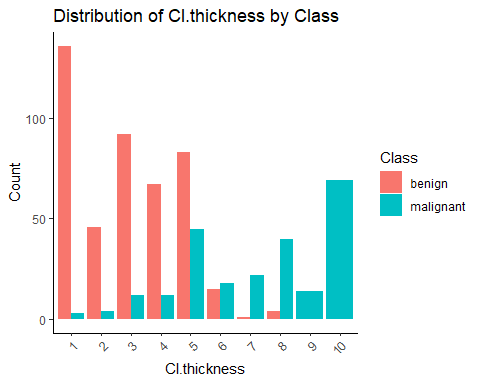
# Count the frequency of each class  
class\_counts <- table(BreastCancer$Class)  
  
# Create a 3D pie chart using plotly  
plot\_ly(labels = c("Benign", "Malignant"),   
 values = class\_counts,   
 type = "pie",   
 marker = list(colors = c("darkblue", "green")),  
 textinfo = "label+percent",  
 textposition = "inside",  
 hole = 0.3) %>%  
 layout(title = "Distribution of Classes in Breast Cancer Dataset",  
 scene = list(camera = list(eye = list(x = 1.25, y = 1.25, z = 1.25))))



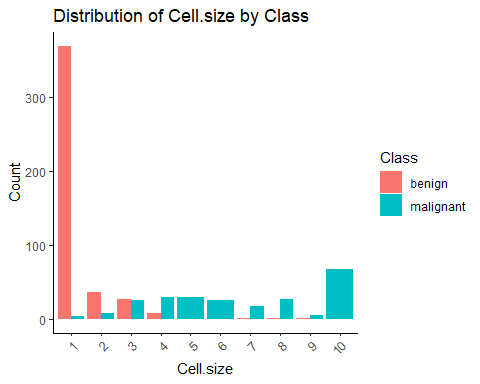
### Distributions of Numeric Variables

# Select factor variables (excluding the 'Class' variable)  
factor\_variables <- BreastCancer[, sapply(BreastCancer, is.factor) & names(BreastCancer) != "Class"]  
  
# Create bar plots for each factor variable  
plots <- lapply(names(factor\_variables), function(var) {  
 ggplot(data = BreastCancer, aes(x = factor\_variables[[var]], fill = as.factor(Class))) +  
 geom\_bar(position = "dodge") +  
 labs(x = var, y = "Count", fill = "Class") +  
 ggtitle(paste("Distribution of", var, "by Class")) +  
 theme\_classic() +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
})  
  
plots

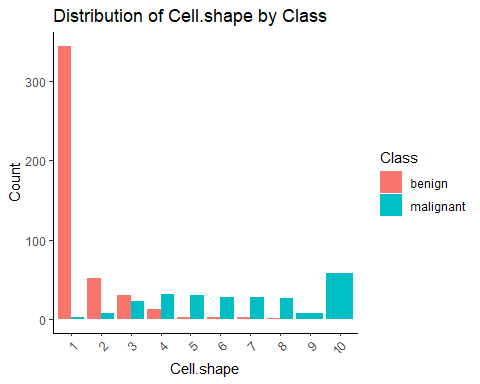
## [[1]]



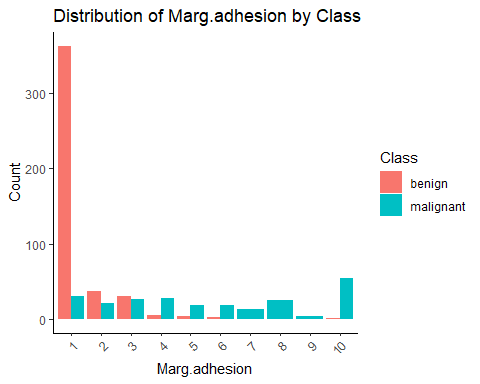
##   
## [[2]]



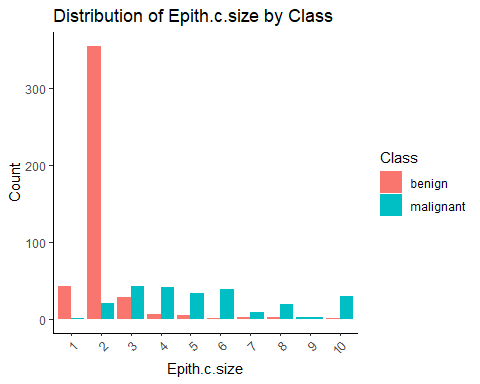
##   
## [[3]]



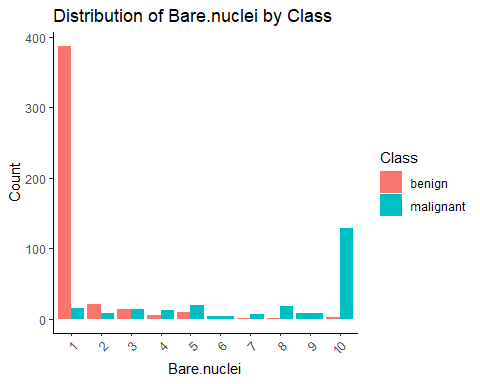
##   
## [[4]]



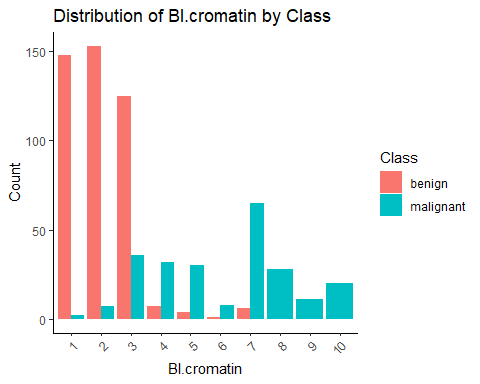
##   
## [[5]]



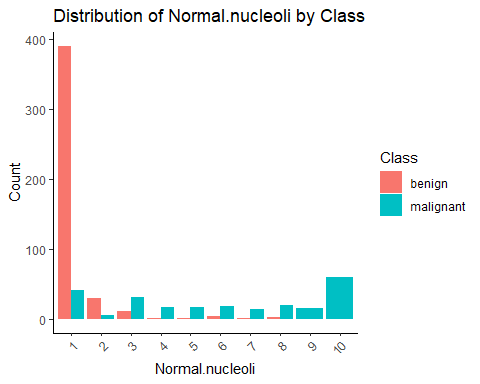
##   
## [[6]]



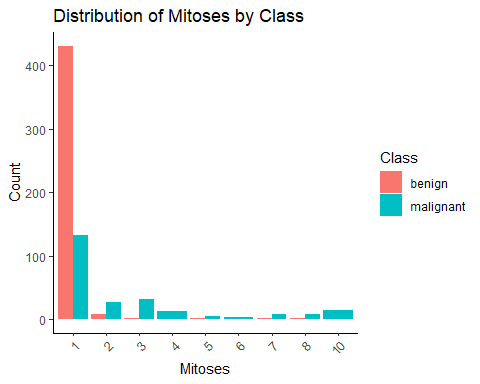
##   
## [[7]]



##   
## [[8]]



##   
## [[9]]



# Splitting the Dataset

# Set the split ratio  
set.seed(2023) # For reproducibility  
ind <- sample.split(BreastCancer$Class, SplitRatio = 0.7)  
  
# Subsetting into Train data  
train <- BreastCancer[ind,]  
cat('The shape of the training dataset:', dim(train))

## The shape of the training dataset: 478 10

# Subsetting into Test data  
test <- BreastCancer[!ind,]  
cat('\nThe shape of the test dataset:', dim(test))

##   
## The shape of the test dataset: 205 10

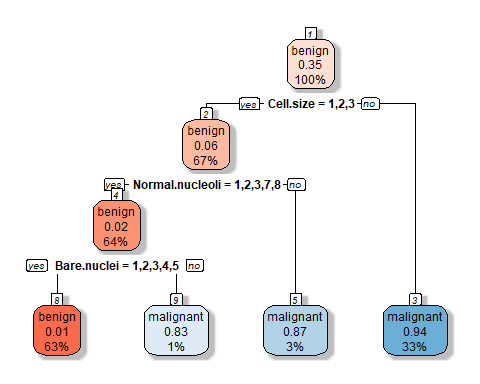
## Decision Tree Classifier

# set seed for reproducibility  
set.seed(2023)  
  
# Train a decision tree classifier  
tree\_model = rpart(Class ~ ., data=train, method="class", minsplit = 10)  
  
# Print the summary of the tree  
print(summary(tree\_model))

## Call:  
## rpart(formula = Class ~ ., data = train, method = "class", minsplit = 10)  
## n= 478   
##   
## CP nsplit rel error xerror xstd  
## 1 0.82634731 0 1.00000000 1.0000000 0.06241774  
## 2 0.06586826 1 0.17365269 0.2215569 0.03498563  
## 3 0.02395210 2 0.10778443 0.1856287 0.03224068  
## 4 0.01000000 3 0.08383234 0.1616766 0.03022315  
##   
## Variable importance  
## Cell.size Cell.shape Bare.nuclei Bl.cromatin Epith.c.size   
## 21 17 15 15 14   
## Marg.adhesion Normal.nucleoli Mitoses Cl.thickness   
## 14 3 1 1   
##   
## Node number 1: 478 observations, complexity param=0.8263473  
## predicted class=benign expected loss=0.3493724 P(node) =1  
## class counts: 311 167  
## probabilities: 0.651 0.349   
## left son=2 (322 obs) right son=3 (156 obs)  
## Primary splits:  
## Cell.size splits as LLLRRRRRRR, improve=162.8326, (0 missing)  
## Cell.shape splits as LLLRRRRRRR, improve=154.2003, (0 missing)  
## Bl.cromatin splits as LLLRRRRRRR, improve=144.0049, (0 missing)  
## Bare.nuclei splits as LLRRRRRRRR, improve=135.2589, (0 missing)  
## Epith.c.size splits as LLRRRRRRRR, improve=132.5151, (0 missing)  
## Surrogate splits:  
## Cell.shape splits as LLLRRRRRRR, agree=0.939, adj=0.814, (0 split)  
## Bl.cromatin splits as LLLRRRRRRR, agree=0.902, adj=0.699, (0 split)  
## Epith.c.size splits as LLRRRRRRRR, agree=0.895, adj=0.679, (0 split)  
## Bare.nuclei splits as LLLRRRRRRR, agree=0.885, adj=0.647, (0 split)  
## Marg.adhesion splits as LLLRRRRRRR, agree=0.881, adj=0.635, (0 split)  
##   
## Node number 2: 322 observations, complexity param=0.06586826  
## predicted class=benign expected loss=0.0621118 P(node) =0.6736402  
## class counts: 302 20  
## probabilities: 0.938 0.062   
## left son=4 (307 obs) right son=5 (15 obs)  
## Primary splits:  
## Normal.nucleoli splits as LLLRRRLLRR, improve=20.36808, (0 missing)  
## Bare.nuclei splits as LLLLRRRRRR, improve=20.18109, (0 missing)  
## Cl.thickness splits as LLLLLLRRRR, improve=16.65518, (0 missing)  
## Bl.cromatin splits as LLLRRLRR--, improve=16.42140, (0 missing)  
## Epith.c.size splits as LLLLRRRRRR, improve=14.17655, (0 missing)  
## Surrogate splits:  
## Mitoses splits as LLRRL-LR-, agree=0.966, adj=0.267, (0 split)  
## Cell.shape splits as LLLLRRRRRR, agree=0.963, adj=0.200, (0 split)  
## Bare.nuclei splits as LLLLLLRRRL, agree=0.963, adj=0.200, (0 split)  
## Cl.thickness splits as LLLLLLRRRR, agree=0.957, adj=0.067, (0 split)  
## Marg.adhesion splits as LLLRRRRRRR, agree=0.957, adj=0.067, (0 split)  
##   
## Node number 3: 156 observations  
## predicted class=malignant expected loss=0.05769231 P(node) =0.3263598  
## class counts: 9 147  
## probabilities: 0.058 0.942   
##   
## Node number 4: 307 observations, complexity param=0.0239521  
## predicted class=benign expected loss=0.0228013 P(node) =0.6422594  
## class counts: 300 7  
## probabilities: 0.977 0.023   
## left son=8 (301 obs) right son=9 (6 obs)  
## Primary splits:  
## Bare.nuclei splits as LLLLLR---R, improve=8.040693, (0 missing)  
## Bl.cromatin splits as LLLLRLRR--, improve=5.263183, (0 missing)  
## Cl.thickness splits as LLLLLLRRRR, improve=5.073916, (0 missing)  
## Epith.c.size splits as LLLLRRRRRR, improve=3.740982, (0 missing)  
## Normal.nucleoli splits as LLR---LL--, improve=2.037805, (0 missing)  
## Surrogate splits:  
## Cl.thickness splits as LLLLLLLLLR, agree=0.987, adj=0.333, (0 split)  
## Marg.adhesion splits as LLLLLLRRRR, agree=0.987, adj=0.333, (0 split)  
## Bl.cromatin splits as LLLLLLLR--, agree=0.984, adj=0.167, (0 split)  
## Mitoses splits as LLR-L-L--, agree=0.984, adj=0.167, (0 split)  
##   
## Node number 5: 15 observations  
## predicted class=malignant expected loss=0.1333333 P(node) =0.03138075  
## class counts: 2 13  
## probabilities: 0.133 0.867   
##   
## Node number 8: 301 observations  
## predicted class=benign expected loss=0.006644518 P(node) =0.6297071  
## class counts: 299 2  
## probabilities: 0.993 0.007   
##   
## Node number 9: 6 observations  
## predicted class=malignant expected loss=0.1666667 P(node) =0.0125523  
## class counts: 1 5  
## probabilities: 0.167 0.833   
##   
## n= 478   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 478 167 benign (0.650627615 0.349372385)   
## 2) Cell.size=1,2,3 322 20 benign (0.937888199 0.062111801)   
## 4) Normal.nucleoli=1,2,3,7,8 307 7 benign (0.977198697 0.022801303)   
## 8) Bare.nuclei=1,2,3,4,5 301 2 benign (0.993355482 0.006644518) \*  
## 9) Bare.nuclei=6,10 6 1 malignant (0.166666667 0.833333333) \*  
## 5) Normal.nucleoli=4,5,6,9,10 15 2 malignant (0.133333333 0.866666667) \*  
## 3) Cell.size=4,5,6,7,8,9,10 156 9 malignant (0.057692308 0.942307692) \*

### Plotting the Tree

##plot the tree  
rpart.plot(tree\_model, box.palette="RdBu", shadow.col="gray", nn=TRUE, yesno = 2)



### Evaluating Decision Tree Classifier

# Make predictions on the test data  
tree\_predictions <- predict(tree\_model, test, type = "class")  
  
# Evaluate the model  
confusion\_matrix <- confusionMatrix(tree\_predictions, test$Class)  
  
# Output the results  
table(tree\_predictions, test$Class)

##   
## tree\_predictions benign malignant  
## benign 129 4  
## malignant 4 68

prop.table(table(tree\_predictions, test$Class),1)

##   
## tree\_predictions benign malignant  
## benign 0.96992481 0.03007519  
## malignant 0.05555556 0.94444444

cat('\n')

cat('\n')

# Confusion Matrix  
cf <- caret::confusionMatrix(data=tree\_predictions,  
 reference=test$Class)  
print(cf)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction benign malignant  
## benign 129 4  
## malignant 4 68  
##   
## Accuracy : 0.961   
## 95% CI : (0.9246, 0.983)  
## No Information Rate : 0.6488   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9144   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.9699   
## Specificity : 0.9444   
## Pos Pred Value : 0.9699   
## Neg Pred Value : 0.9444   
## Prevalence : 0.6488   
## Detection Rate : 0.6293   
## Detection Prevalence : 0.6488   
## Balanced Accuracy : 0.9572   
##   
## 'Positive' Class : benign   
##

The Decision Tree model was evaluated using a confusion matrix. The confusion matrix shows the number of true positives, true negatives, false positives, and false negatives. The model predicted 129 cases as benign and they were actually benign, while 4 cases were predicted as benign but were actually malignant. On the other hand, the model predicted 68 cases as malignant and they were actually malignant, while 4 cases were predicted as malignant but were actually benign.

The accuracy of the model is 0.961, which means that it correctly classified 96.1% of the cases. The sensitivity (also known as true positive rate) is 0.9699, indicating that the model correctly identified 96.99% of the malignant cases. The specificity (also known as true negative rate) is 0.9444, indicating that the model correctly identified 94.44% of the benign cases. The positive predictive value (also known as precision) is 0.9699, indicating that when the model predicted a case as malignant, it was correct 96.99% of the time. The negative predictive value is 0.9444, indicating that when the model predicted a case as benign, it was correct 94.44% of the time.

## Support Vector Machine

### Checking for Best Parameters

# set seed for reproducibility  
set.seed(2023)  
  
# create svm model  
svm\_model <- tune.svm(Class~ Cl.thickness +   
 Cell.size +   
 Cell.shape +   
 Marg.adhesion +   
 Epith.c.size +   
 Bare.nuclei +   
 Bl.cromatin +   
 Normal.nucleoli +   
 Mitoses,   
 data = train, gamma = 10^(-6:-1), cost = 10^(-1:1))  
  
summary(svm\_model)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## gamma cost  
## 0.1 0.1  
##   
## - best performance: 0.02925532   
##   
## - Detailed performance results:  
## gamma cost error dispersion  
## 1 1e-06 0.1 0.34942376 0.07353231  
## 2 1e-05 0.1 0.34942376 0.07353231  
## 3 1e-04 0.1 0.34942376 0.07353231  
## 4 1e-03 0.1 0.34942376 0.07353231  
## 5 1e-02 0.1 0.21764184 0.07973222  
## 6 1e-01 0.1 0.02925532 0.01465375  
## 7 1e-06 1.0 0.34942376 0.07353231  
## 8 1e-05 1.0 0.34942376 0.07353231  
## 9 1e-04 1.0 0.34942376 0.07353231  
## 10 1e-03 1.0 0.15270390 0.06376075  
## 11 1e-02 1.0 0.03554965 0.01976131  
## 12 1e-01 1.0 0.03138298 0.02027389  
## 13 1e-06 10.0 0.34942376 0.07353231  
## 14 1e-05 10.0 0.34942376 0.07353231  
## 15 1e-04 10.0 0.14645390 0.05825869  
## 16 1e-03 10.0 0.03554965 0.01976131  
## 17 1e-02 10.0 0.03351064 0.02256356  
## 18 1e-01 10.0 0.03138298 0.01773641

### Support Vector Machine with Best Parameters

# set seed for reproducibility  
set.seed(2023)  
  
# Create an SVM model  
svm\_model2 <- svm(Class~ Cl.thickness +   
 Cell.size +   
 Cell.shape +   
 Marg.adhesion +   
 Epith.c.size +   
 Bare.nuclei +   
 Bl.cromatin +   
 Normal.nucleoli +   
 Mitoses,   
 data = train, type = 'C-classification', gamma = 0.1, cost = 0.1)  
  
summary(svm\_model2)

##   
## Call:  
## svm(formula = Class ~ Cl.thickness + Cell.size + Cell.shape + Marg.adhesion +   
## Epith.c.size + Bare.nuclei + Bl.cromatin + Normal.nucleoli +   
## Mitoses, data = train, type = "C-classification", gamma = 0.1,   
## cost = 0.1)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 0.1   
##   
## Number of Support Vectors: 215  
##   
## ( 104 111 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## benign malignant

### Predictions Using SVM With Best Parameters

# Remove the 'Class' column (labels) from the test dataset  
test\_features <- test[, -which(names(test) == "Class")]  
  
# Make predictions using the SVM model and the test features  
svm\_predictions <- predict(svm\_model2, newdata = test\_features)  
  
# Output the results  
table(svm\_predictions, test$Class)

##   
## svm\_predictions benign malignant  
## benign 125 2  
## malignant 8 70

prop.table(table(svm\_predictions, test$Class),1)

##   
## svm\_predictions benign malignant  
## benign 0.98425197 0.01574803  
## malignant 0.10256410 0.89743590

cat('\n')

cat('\n')

# Confusion Matrix  
cf <- caret::confusionMatrix(data=svm\_predictions,  
 reference=test$Class)  
print(cf)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction benign malignant  
## benign 125 2  
## malignant 8 70  
##   
## Accuracy : 0.9512   
## 95% CI : (0.9121, 0.9764)  
## No Information Rate : 0.6488   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.895   
##   
## Mcnemar's Test P-Value : 0.1138   
##   
## Sensitivity : 0.9398   
## Specificity : 0.9722   
## Pos Pred Value : 0.9843   
## Neg Pred Value : 0.8974   
## Prevalence : 0.6488   
## Detection Rate : 0.6098   
## Detection Prevalence : 0.6195   
## Balanced Accuracy : 0.9560   
##   
## 'Positive' Class : benign   
##

The SVM (Support Vector Machine) model was evaluated using a confusion matrix. The model predicted 125 cases as benign and they were actually benign, while 2 cases were predicted as benign but were actually malignant. On the other hand, the model predicted 70 cases as malignant and they were actually malignant, while 8 cases were predicted as malignant but were actually benign.

The accuracy of the model is 0.9512, which means that it correctly classified 95.12% of the cases. The sensitivity (also known as true positive rate) is 0.9398, indicating that the model correctly identified 93.98% of the malignant cases. The specificity (also known as true negative rate) is 0.9722, indicating that the model correctly identified 97.22% of the benign cases. The positive predictive value (also known as precision) is 0.9843, indicating that when the model predicted a case as malignant, it was correct 98.43% of the time. The negative predictive value is 0.8974, indicating that when the model predicted a case as benign, it was correct 89.74% of the time.

## XGBOOST Model

# Convert the class labels to 0 and 1 for binary classification  
train$Class <- ifelse(train$Class == "benign", 0, 1)  
test$Class <- ifelse(test$Class == "benign", 0, 1)  
  
# Convert entire train and test datasets to numeric  
train <- as.data.frame(lapply(train, as.numeric))  
test <- as.data.frame(lapply(test, as.numeric))  
  
# Convert the training and test data to DMatrix format  
dtrain <- xgb.DMatrix(data = as.matrix(train[, -which(names(train) == "Class")]), label = train$Class)  
dtest <- xgb.DMatrix(data = as.matrix(test[, -which(names(test) == "Class")]), label = test$Class)  
  
# Define XGBoost parameters  
params <- list(  
 # Binary classification problem  
 objective = "binary:logistic",   
   
 # Evaluation metric (logarithmic loss)  
 eval\_metric = "logloss",   
   
 # Learning rate  
 eta = 0.3,   
   
 # Maximum depth of trees  
 max\_depth = 6,   
   
 # Minimum sum of instance weight needed in a child  
 min\_child\_weight = 1,   
   
 # Subsample ratio of the training data  
 subsample = 1,   
   
 # Subsample ratio of columns when constructing each tree  
 colsample\_bytree = 1   
)  
  
set.seed(2023)  
# Train the XGBoost model  
xgb\_model <- xgboost(data = dtrain, params = params, nrounds = 100, verbose = 1)

## [1] train-logloss:0.465075   
## [2] train-logloss:0.338282   
## [3] train-logloss:0.258116   
## [4] train-logloss:0.200511   
## [5] train-logloss:0.159202   
## [6] train-logloss:0.128892   
## [7] train-logloss:0.105181   
## [8] train-logloss:0.087831   
## [9] train-logloss:0.074789   
## [10] train-logloss:0.063867   
## [11] train-logloss:0.054624   
## [12] train-logloss:0.048601   
## [13] train-logloss:0.043928   
## [14] train-logloss:0.040212   
## [15] train-logloss:0.036630   
## [16] train-logloss:0.033789   
## [17] train-logloss:0.031420   
## [18] train-logloss:0.028590   
## [19] train-logloss:0.026897   
## [20] train-logloss:0.025220   
## [21] train-logloss:0.024247   
## [22] train-logloss:0.023204   
## [23] train-logloss:0.022476   
## [24] train-logloss:0.021763   
## [25] train-logloss:0.020993   
## [26] train-logloss:0.020269   
## [27] train-logloss:0.019671   
## [28] train-logloss:0.019168   
## [29] train-logloss:0.018784   
## [30] train-logloss:0.018452   
## [31] train-logloss:0.018096   
## [32] train-logloss:0.017705   
## [33] train-logloss:0.017157   
## [34] train-logloss:0.016868   
## [35] train-logloss:0.016452   
## [36] train-logloss:0.016134   
## [37] train-logloss:0.015867   
## [38] train-logloss:0.015636   
## [39] train-logloss:0.015463   
## [40] train-logloss:0.015221   
## [41] train-logloss:0.015095   
## [42] train-logloss:0.014997   
## [43] train-logloss:0.014827   
## [44] train-logloss:0.014527   
## [45] train-logloss:0.014313   
## [46] train-logloss:0.014222   
## [47] train-logloss:0.014103   
## [48] train-logloss:0.013938   
## [49] train-logloss:0.013832   
## [50] train-logloss:0.013600   
## [51] train-logloss:0.013458   
## [52] train-logloss:0.013289   
## [53] train-logloss:0.013146   
## [54] train-logloss:0.013058   
## [55] train-logloss:0.012968   
## [56] train-logloss:0.012802   
## [57] train-logloss:0.012622   
## [58] train-logloss:0.012466   
## [59] train-logloss:0.012384   
## [60] train-logloss:0.012326   
## [61] train-logloss:0.012186   
## [62] train-logloss:0.012110   
## [63] train-logloss:0.012016   
## [64] train-logloss:0.011936   
## [65] train-logloss:0.011874   
## [66] train-logloss:0.011831   
## [67] train-logloss:0.011757   
## [68] train-logloss:0.011578   
## [69] train-logloss:0.011422   
## [70] train-logloss:0.011382   
## [71] train-logloss:0.011322   
## [72] train-logloss:0.011255   
## [73] train-logloss:0.011142   
## [74] train-logloss:0.011103   
## [75] train-logloss:0.011047   
## [76] train-logloss:0.010952   
## [77] train-logloss:0.010833   
## [78] train-logloss:0.010797   
## [79] train-logloss:0.010727   
## [80] train-logloss:0.010647   
## [81] train-logloss:0.010612   
## [82] train-logloss:0.010551   
## [83] train-logloss:0.010482   
## [84] train-logloss:0.010448   
## [85] train-logloss:0.010348   
## [86] train-logloss:0.010283   
## [87] train-logloss:0.010206   
## [88] train-logloss:0.010177   
## [89] train-logloss:0.010127   
## [90] train-logloss:0.010094   
## [91] train-logloss:0.010050   
## [92] train-logloss:0.009998   
## [93] train-logloss:0.009940   
## [94] train-logloss:0.009907   
## [95] train-logloss:0.009860   
## [96] train-logloss:0.009809   
## [97] train-logloss:0.009778   
## [98] train-logloss:0.009736   
## [99] train-logloss:0.009691   
## [100] train-logloss:0.009662

# Make predictions on the test data  
xgb\_predictions <- predict(xgb\_model, dtest)  
  
# Convert predictions to class labels (0 or 1)  
xgb\_predictions <- ifelse(xgb\_predictions > 0.5, 1, 0)  
  
# Calculate accuracy  
accuracy <- sum(xgb\_predictions == test$Class) / nrow(test)  
print(paste("Accuracy:", accuracy))

## [1] "Accuracy: 0.970731707317073"

### Evaluation of the XGBOOST Metrics

# Convert predictions and true labels to factors with levels "benign" and "malignant"  
predicted\_labels <- factor(ifelse(xgb\_predictions == 0, "benign", "malignant"), levels = c("benign", "malignant"))  
test$Class <- factor(ifelse(test$Class == 0, "benign", "malignant"), levels = c("benign", "malignant"))  
  
# Create confusion matrix  
confusion\_matrix <- confusionMatrix(predicted\_labels, test$Class)  
  
# Output the results  
# Output the results  
table(predicted\_labels, test$Class)

##   
## predicted\_labels benign malignant  
## benign 132 5  
## malignant 1 67

prop.table(table(predicted\_labels, test$Class),1)

##   
## predicted\_labels benign malignant  
## benign 0.96350365 0.03649635  
## malignant 0.01470588 0.98529412

cat('\n')

cat('\n')

# Confusion Matrix  
cf <- caret::confusionMatrix(data=predicted\_labels,  
 reference=test$Class)  
print(cf)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction benign malignant  
## benign 132 5  
## malignant 1 67  
##   
## Accuracy : 0.9707   
## 95% CI : (0.9374, 0.9892)  
## No Information Rate : 0.6488   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9349   
##   
## Mcnemar's Test P-Value : 0.2207   
##   
## Sensitivity : 0.9925   
## Specificity : 0.9306   
## Pos Pred Value : 0.9635   
## Neg Pred Value : 0.9853   
## Prevalence : 0.6488   
## Detection Rate : 0.6439   
## Detection Prevalence : 0.6683   
## Balanced Accuracy : 0.9615   
##   
## 'Positive' Class : benign   
##

The XGBoost model was evaluated using a confusion matrix. The model predicted 132 cases as benign and they were actually benign, while 5 cases were predicted as benign but were actually malignant. On the other hand, the model predicted 67 cases as malignant and they were actually malignant, while 1 case was predicted as malignant but was actually benign.

The accuracy of the model is 0.9707, which means that it correctly classified 97.07% of the cases. The sensitivity (also known as true positive rate) is 0.9925, indicating that the model correctly identified 99.25% of the malignant cases. The specificity (also known as true negative rate) is 0.9306, indicating that the model correctly identified 93.06% of the benign cases. The positive predictive value (also known as precision) is 0.9635, indicating that when the model predicted a case as malignant, it was correct 96.35% of the time. The negative predictive value is 0.9853, indicating that when the model predicted a case as benign, it was correct 98.53% of the time.

## Comparison of Decision Tree, SVM, and XGBoost.

Decision Tree: \* Accuracy: 0.961 \* Sensitivity: 0.9699 \* Specificity: 0.9444

SVM: \* Accuracy: 0.9512 \* Sensitivity: 0.9398 \* Specificity: 0.9722

XGBoost: \* Accuracy: 0.9707 \* Sensitivity: 0.9925 \* Specificity: 0.9306

Based on these metrics, the XGBoost model performed the best among the three models. It achieved the highest accuracy (0.9707) and sensitivity (0.9925), indicating that it correctly classified the majority of cases and had a low rate of false negatives. However, it had a slightly lower specificity (0.9306) compared to the SVM model. Overall, the XGBoost model demonstrated a good balance between accuracy and sensitivity, making it the best-performing model in this comparison.