Breast Cancer - An Ensemble Approach For Classifying Tumors

David Pershall

9/27/2021

Introduction

This data set is provided by the UCI Machine Learning repository at the following web address.

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

The data set description gives us some helpful information; the id number of the sample, the diagnosis, and 30 features.

The features are computed from a digitized image of a fine needle aspirate of a breast mass. They describe characteristics of the cell nuclei present in the image. There are ten real-valued features for each cell nucleus.

They are:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness (perimeter^2 / area - 1.0)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension ("coastline approximation" - 1)

The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in a total of 30 measured features.

The goal of this project is to analyze the features and compose an algorithm for classifying the samples as "M" for malignant or "B" for benign with as high a rate of accuracy as possible. Let's see if we can make it easier for doctors to determine if the fine needle aspirate of a mass is indeed malignant.

Libraries

These are the libraries used in my project.

```
library(matrixStats)
library(tidyverse)
library(caret)
library(gam)
library(rattle)
library(ggrepel)
```

Data Pull

The following pulls the data from the machine learning database at UCI. The feature names are not included in the data file, so we have to add them manually.

```
temp_dat <- read.csv(url(
  "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data"),
                     header = FALSE,
                     col.names = c("ID", "diagnosis", "radius_mean",
                                   "texture_mean", "perimeter_mean",
                                   "area_mean", "smoothness_mean",
                                   "compactness_mean", "concavity_mean",
                                   "concave_pts_mean", "symmetry_mean",
                                   "fractal_dim_mean", "radius_se",
                                   "texture_se", "perimeter_se", "area_se",
                                   "smoothness_se", "compactness_se",
                                   "concavity_se", "concave_pts_se",
                                   "symmetry_se", "fractal_dim_se",
                                   "radius_worst", "texture_worst",
                                   "perimeter_worst", "area_worst",
                                   "smoothness_worst", "compactness_worst",
                                   "concavity_worst", "concave_pts_worst",
                                   "symmetry_worst", "fractal_dim_worst"))

# make the diagnosis a factor B or M
temp_dat <- temp_dat %>% mutate(diagnosis = as.factor(diagnosis))

# take a quick look at the data
glimpse(temp_dat)
```

```
## Rows: 569
## Columns: 32
## $ ID                <int> 842302, 842517, 84300903, 84348301, 84358402, 843786~
## $ diagnosis         <fct> M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M~
## $ radius_mean       <dbl> 17.990, 20.570, 19.690, 11.420, 20.290, 12.450, 18.2~
## $ texture_mean      <dbl> 10.38, 17.77, 21.25, 20.38, 14.34, 15.70, 19.98, 20.~
## $ perimeter_mean    <dbl> 122.80, 132.90, 130.00, 77.58, 135.10, 82.57, 119.60~
## $ area_mean         <dbl> 1001.0, 1326.0, 1203.0, 386.1, 1297.0, 477.1, 1040.0~
## $ smoothness_mean   <dbl> 0.11840, 0.08474, 0.10960, 0.14250, 0.10030, 0.12780~
## $ compactness_mean  <dbl> 0.27760, 0.07864, 0.15990, 0.28390, 0.13280, 0.17000~
## $ concavity_mean    <dbl> 0.30010, 0.08690, 0.19740, 0.24140, 0.19800, 0.15780~
## $ concave_pts_mean  <dbl> 0.14710, 0.07017, 0.12790, 0.10520, 0.10430, 0.08089~
## $ symmetry_mean     <dbl> 0.2419, 0.1812, 0.2069, 0.2597, 0.1809, 0.2087, 0.17~
## $ fractal_dim_mean  <dbl> 0.07871, 0.05667, 0.05999, 0.09744, 0.05883, 0.07613~
## $ radius_se         <dbl> 1.0950, 0.5435, 0.7456, 0.4956, 0.7572, 0.3345, 0.44~
## $ texture_se        <dbl> 0.9053, 0.7339, 0.7869, 1.1560, 0.7813, 0.8902, 0.77~
## $ perimeter_se      <dbl> 8.589, 3.398, 4.585, 3.445, 5.438, 2.217, 3.180, 3.8~
## $ area_se           <dbl> 153.40, 74.08, 94.03, 27.23, 94.44, 27.19, 53.91, 50~
## $ smoothness_se     <dbl> 0.006399, 0.005225, 0.006150, 0.009110, 0.011490, 0.~
## $ compactness_se    <dbl> 0.049040, 0.013080, 0.040060, 0.074580, 0.024610, 0.~
## $ concavity_se      <dbl> 0.05373, 0.01860, 0.03832, 0.05661, 0.05688, 0.03672~
## $ concave_pts_se    <dbl> 0.015870, 0.013400, 0.020580, 0.018670, 0.018850, 0.~
## $ symmetry_se       <dbl> 0.03003, 0.01389, 0.02250, 0.05963, 0.01756, 0.02165~
## $ fractal_dim_se    <dbl> 0.006193, 0.003532, 0.004571, 0.009208, 0.005115, 0.~
## $ radius_worst      <dbl> 25.38, 24.99, 23.57, 14.91, 22.54, 15.47, 22.88, 17.~
```

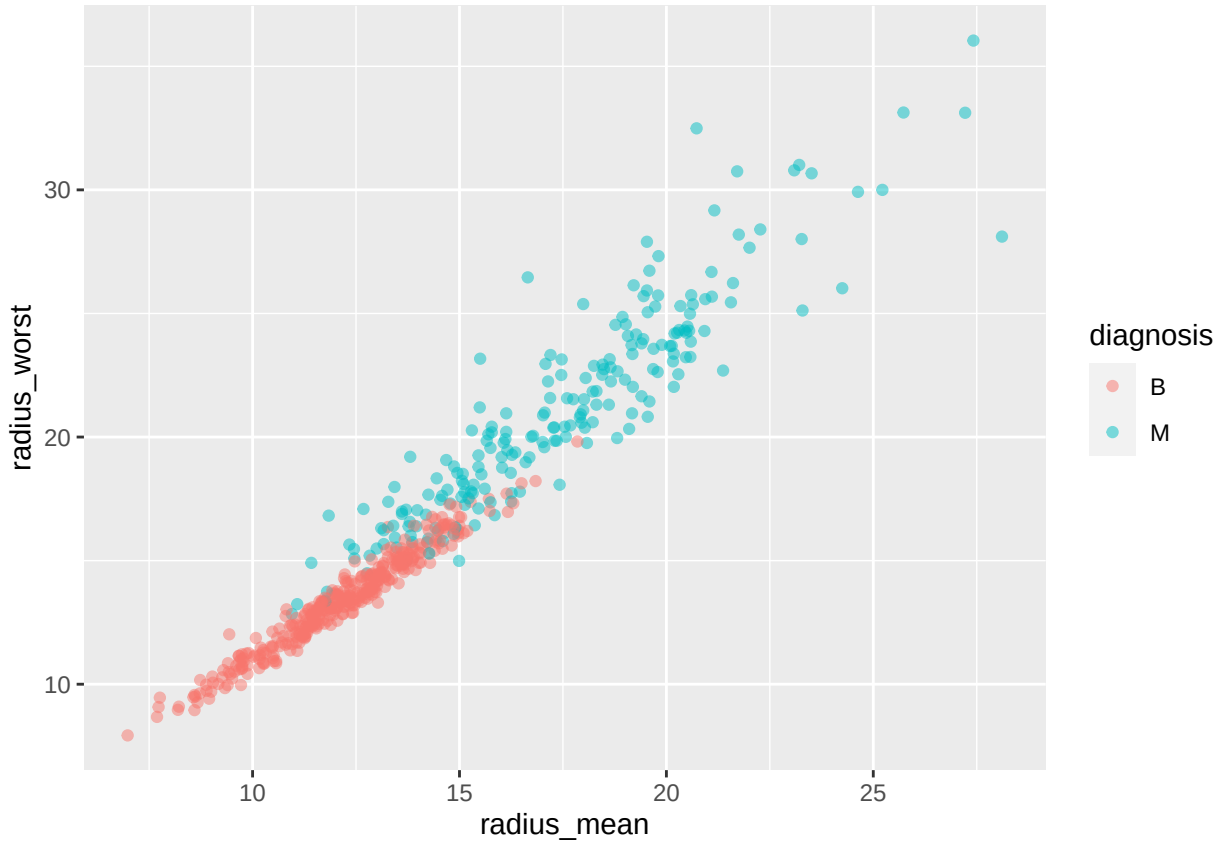
```
## $ texture_worst     <dbl> 17.33, 23.41, 25.53, 26.50, 16.67, 23.75, 27.66, 28.~
## $ perimeter_worst   <dbl> 184.60, 158.80, 152.50, 98.87, 152.20, 103.40, 153.2~
## $ area_worst        <dbl> 2019.0, 1956.0, 1709.0, 567.7, 1575.0, 741.6, 1606.0~
## $ smoothness_worst  <dbl> 0.1622, 0.1238, 0.1444, 0.2098, 0.1374, 0.1791, 0.14~
## $ compactness_worst <dbl> 0.6656, 0.1866, 0.4245, 0.8663, 0.2050, 0.5249, 0.25~
## $ concavity_worst   <dbl> 0.71190, 0.24160, 0.45040, 0.68690, 0.40000, 0.53550~
## $ concave_pts_worst <dbl> 0.26540, 0.18600, 0.24300, 0.25750, 0.16250, 0.17410~
## $ symmetry_worst    <dbl> 0.4601, 0.2750, 0.3613, 0.6638, 0.2364, 0.3985, 0.30~
## $ fractal_dim_worst <dbl> 0.11890, 0.08902, 0.08758, 0.17300, 0.07678, 0.12440~
```

OK, it looks like the data pull went according to plan.

Exploration

Let's start off with a relatively simple scatter plot comparing the perimeter mean and the perimeter worst features.

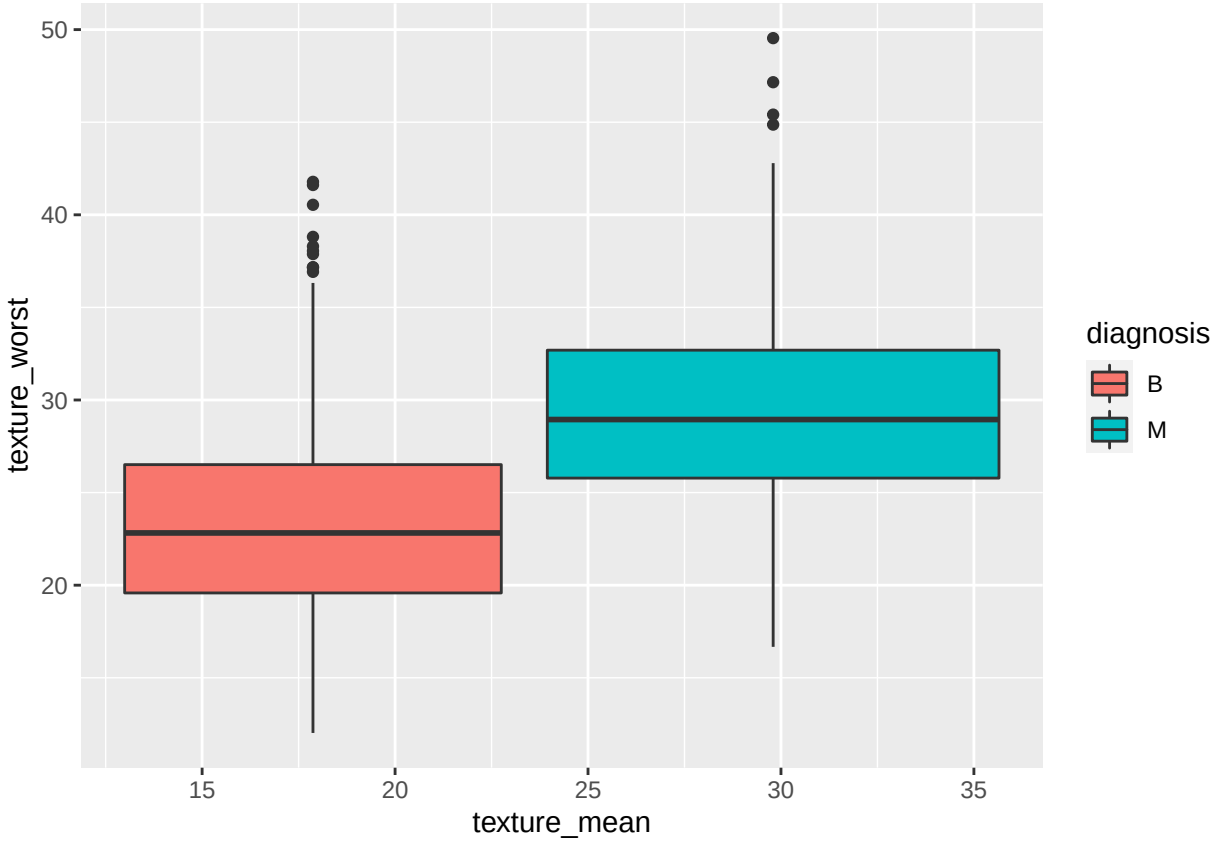
```
temp_dat %>%
  group_by(diagnosis) %>%
  ggplot(aes(radius_mean, radius_worst, color = diagnosis)) +
  geom_point(alpha = .5)
```



This gives us some helpful information. First, it tells us that the average radius for benign tumors is lower than the average radius for malignant tumors. Secondly, it shows us that there is some overlap where we could potentially misdiagnose the tumors if these were the only features measured.

Let's examine some of the other features as well. The following code makes a box plot of the texture mean and texture worst features.

```
temp_dat %>% group_by(diagnosis) %>%
  ggplot(aes(texture_mean, texture_worst, fill = diagnosis)) +
  geom_boxplot()
```



This shows us that, on average, benign tumors have lower values of both texture mean and texture worst measurements. It also shows us that some samples would cause an error in classification if we did not know their diagnosis in advance. So, we need to dig a little deeper and see if we can get a better delineation of classes (benign or malignant) through principal component analysis.

PCA

The first step will be to coerce the data into a list of two variables. The first variable will be termed x and it will hold all the features without the ID or diagnosis, and the second variable, y, will hold only the diagnosis.

```
dat <- list(x = as.matrix(temp_dat[3:32]), y=as.factor(temp_dat$diagnosis))

# Now I will check the class of the two variables and the dimensions of the
# x feature matrix to ensure all the features I wanted to keep have been
# maintained
class(dat$x)
```

```
## [1] "matrix" "array"
```

```
class(dat$y)
```

```
## [1] "factor"
```

```r
# dimensions
dim(dat$x)[1]
```

```
## [1] 569
```

```r
dim(dat$x)[2]
```

```
## [1] 30
```

The data is now stored in "dat" and the "dat$x" variable has maintained all 569 samples along with the 30 feature observations.

Time to run some calculations.

First up, I would like to know what proportion of the samples have been classified as malignant, and what proportion of samples have been classified as benign?

```r
mean(dat$y == "M")
```

```
## [1] 0.3725835
```

```r
mean(dat$y == "B")
```

```
## [1] 0.6274165
```

I wonder what feature has the highest mean, and what feature has the lowest standard deviation?

```r
which.max(colMeans(dat$x))
```

```
## area_worst
##         24
```

```r
which.min(colSds(dat$x))
```

```
## [1] 20
```

The area worst feature has the highest average measurement, which makes sense. Column 20 corresponds with the standard error of the fractal dimension ("coastline approximation" - 1), and it has the lowest standard deviation.

Next up, I want to center and then scale the features.

```r
x_centered <- sweep(dat$x, 2, colMeans(dat$x))
x_scaled <- sweep(x_centered, 2, colSds(dat$x), FUN = "/")

sd(x_scaled[,1])
```

```
## [1] 1
```

```r
median(x_scaled[,1])
```

```
## [1] -0.2148925
```

Let's use the x_scaled data to calculate the distance between all samples. Then we can compute the average distance between the first sample, which is malignant, and the other malignant samples; followed by the average distance between the first malignant sample and other benign samples.

```r
# average distance between all samples
d_samples <- dist(x_scaled)

# first malignant to other malignant samples - average distance
dist_MtoM <- as.matrix(d_samples)[1, dat$y == "M"]
mean(dist_MtoM[2:length(dist_MtoM)])
```

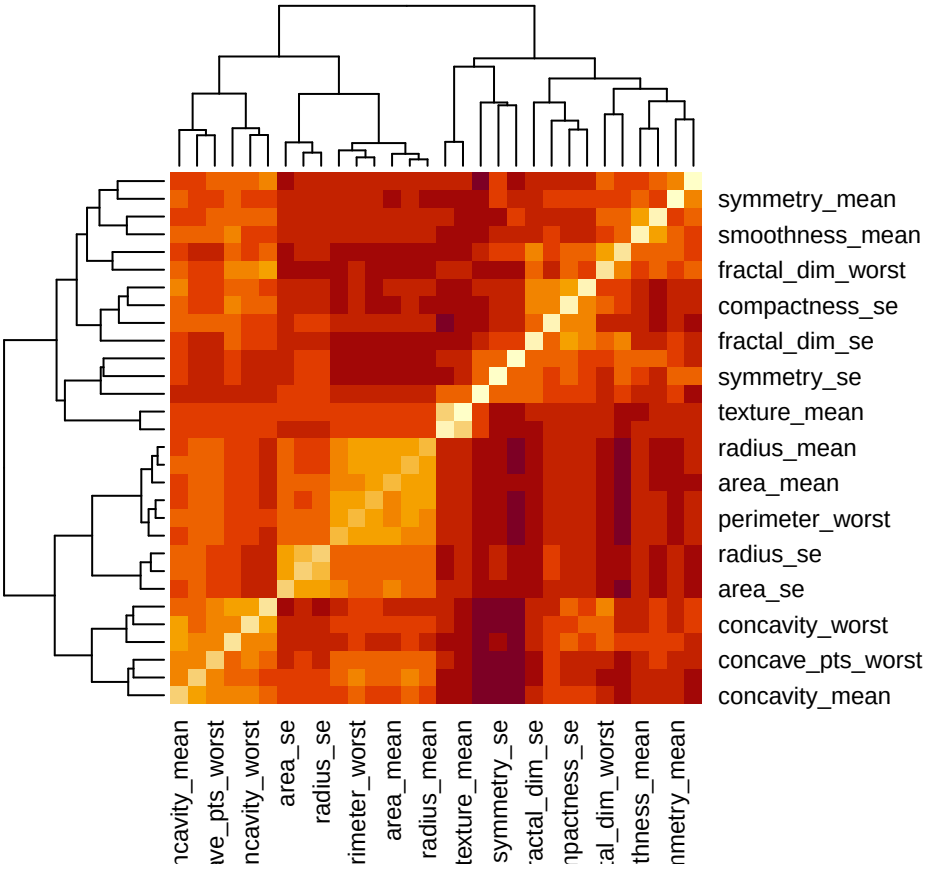
```
## [1] 9.50695
```

```
# first malignant to other benign samples - average distance
dist_MtoB <- as.matrix(d_samples)[1, dat$y == "B"]
mean(dist_MtoB)
```

```
## [1] 13.14004
```

Now, I will make a heat-map of the relationship between features using the scaled matrix.

```
d_features <- dist(t(x_scaled))
heatmap(as.matrix(d_features))
```



We can clearly see that there are high correlations between some of our features.

Let's cut the features into 5 groups and examine the hierarchy of the features.

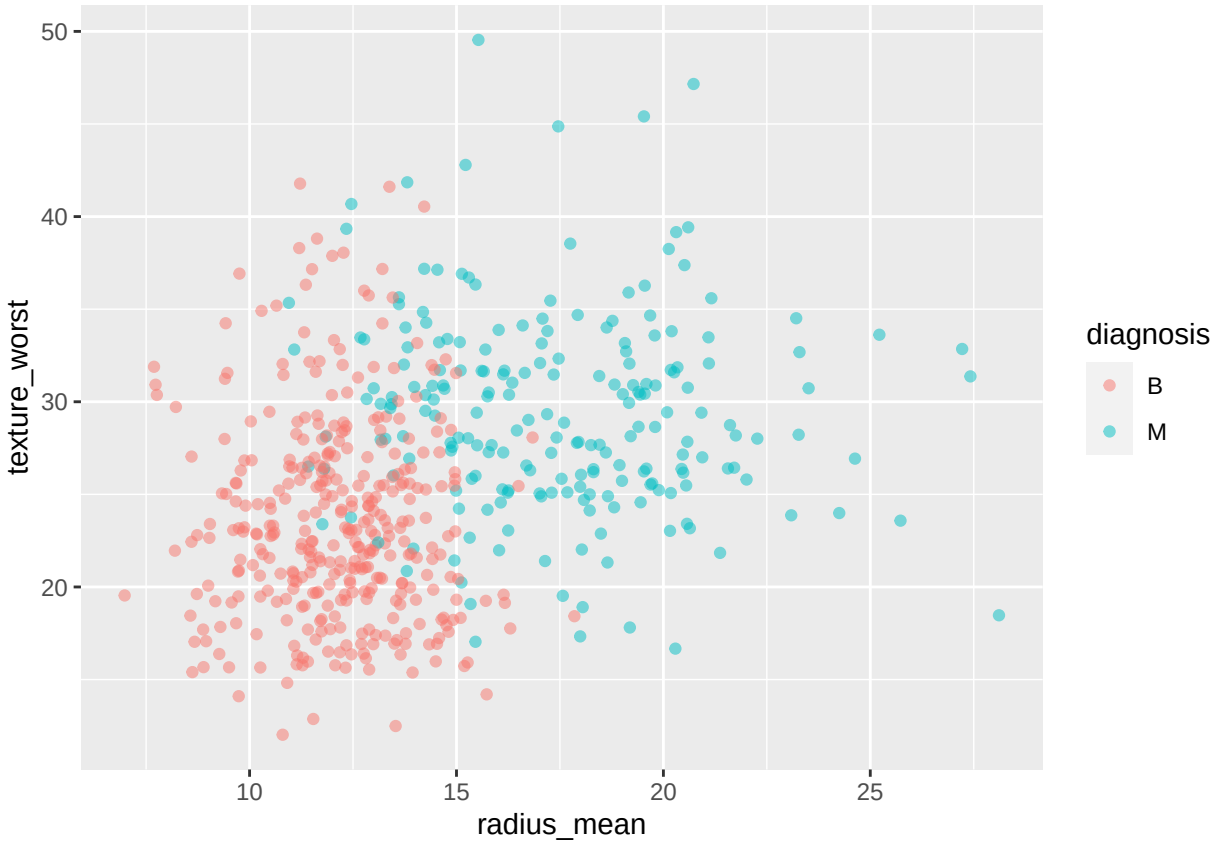
```
h <- hclust(d_features)
groups <- cutree(h, k = 5)
split(names(groups), groups)
```

```
## $`1`
##  [1] "radius_mean"       "perimeter_mean"    "area_mean"
##  [4] "concavity_mean"    "concave_pts_mean"  "radius_se"
##  [7] "perimeter_se"      "area_se"           "radius_worst"
## [10] "perimeter_worst"   "area_worst"        "concave_pts_worst"
##
## $`2`
## [1] "texture_mean"  "texture_worst"
##
```

```
## $`3`
## [1] "smoothness_mean"   "compactness_mean"  "symmetry_mean"
## [4] "fractal_dim_mean"  "smoothness_worst"  "compactness_worst"
## [7] "concavity_worst"   "symmetry_worst"    "fractal_dim_worst"
##
## $`4`
## [1] "texture_se"    "smoothness_se" "symmetry_se"
##
## $`5`
## [1] "compactness_se" "concavity_se"   "concave_pts_se" "fractal_dim_se"
```

We can use these groups to attempt a delineation between benign and malignant tumors. The following code selects one feature from the 1st group and one feature from the 2nd group and makes a scatter plot.

```
temp_dat %>% ggplot(aes(radius_mean, texture_worst, color = diagnosis)) +
  geom_point(alpha = 0.5)
```



OK, we are getting closer. However, there is still quite a bit of overlap between benign and malignant tumors.

I will use principal component analysis to discover better lines of delineation between the classes of tumor growth. Let's use the scaled version of the features for the analysis.

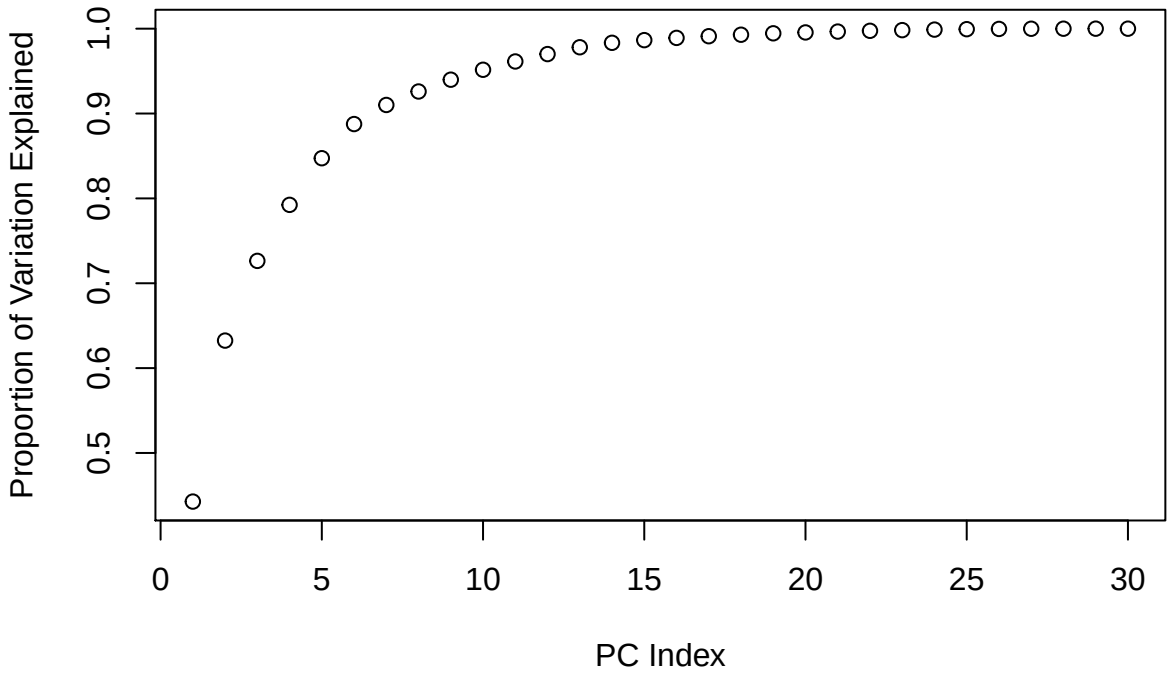
```
pca <- prcomp(x_scaled)
pca_sum <- summary(pca)

# show the summary of the principal component analysis
pca_sum$importance
```

```
##                             PC1       PC2       PC3       PC4       PC5       PC6
```

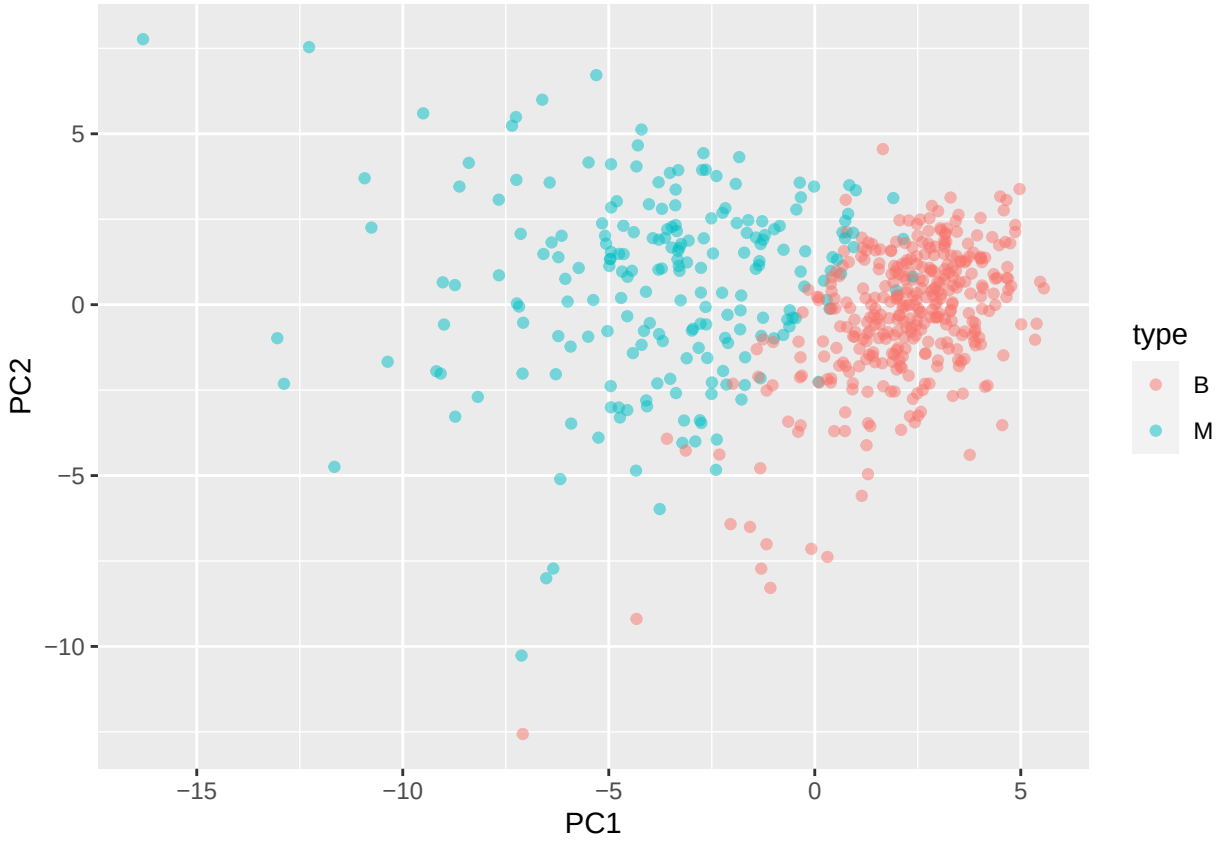
```
## Standard deviation     3.644394 2.385656 1.678675 1.407352 1.284029 1.098798
## Proportion of Variance 0.442720 0.189710 0.093930 0.066020 0.054960 0.040250
## Cumulative Proportion  0.442720 0.632430 0.726360 0.792390 0.847340 0.887590
##                            PC7       PC8       PC9      PC10      PC11
## Standard deviation     0.8217178 0.6903746 0.6456739 0.5921938 0.5421399
## Proportion of Variance 0.0225100 0.0158900 0.0139000 0.0116900 0.0098000
## Cumulative Proportion  0.9101000 0.9259800 0.9398800 0.9515700 0.9613700
##                           PC12      PC13      PC14      PC15      PC16
## Standard deviation     0.5110395 0.4912815 0.3962445 0.3068142 0.2826001
## Proportion of Variance 0.0087100 0.0080500 0.0052300 0.0031400 0.0026600
## Cumulative Proportion  0.9700700 0.9781200 0.9833500 0.9864900 0.9891500
##                           PC17      PC18      PC19      PC20      PC21
## Standard deviation     0.2437192 0.2293878 0.2224356 0.1765203 0.1731268
## Proportion of Variance 0.0019800 0.0017500 0.0016500 0.0010400 0.0010000
## Cumulative Proportion  0.9911300 0.9928800 0.9945300 0.9955700 0.9965700
##                           PC22      PC23      PC24      PC25      PC26
## Standard deviation     0.1656484 0.1560155 0.1343689 0.1244238 0.0904303
## Proportion of Variance 0.0009100 0.0008100 0.0006000 0.0005200 0.0002700
## Cumulative Proportion  0.9974900 0.9983000 0.9989000 0.9994200 0.9996900
##                           PC27      PC28       PC29       PC30
## Standard deviation     0.08306903 0.0398665 0.02736427 0.01153451
## Proportion of Variance 0.00023000 0.0000500 0.00002000 0.00000000
## Cumulative Proportion  0.99992000 0.9999700 1.00000000 1.00000000
```

```r
# plot the proportion of variance explained and the principal component index
var_explained <- cumsum(pca$sdev^2/sum(pca$sdev^2))
plot(var_explained, ylab = "Proportion of Variation Explained", xlab = "PC Index")
```



Here we see that 90% of the variance is explained by the first 7 principal components alone. Let's make a quick scatter plot of the first two principal components with color representing the tumor type.

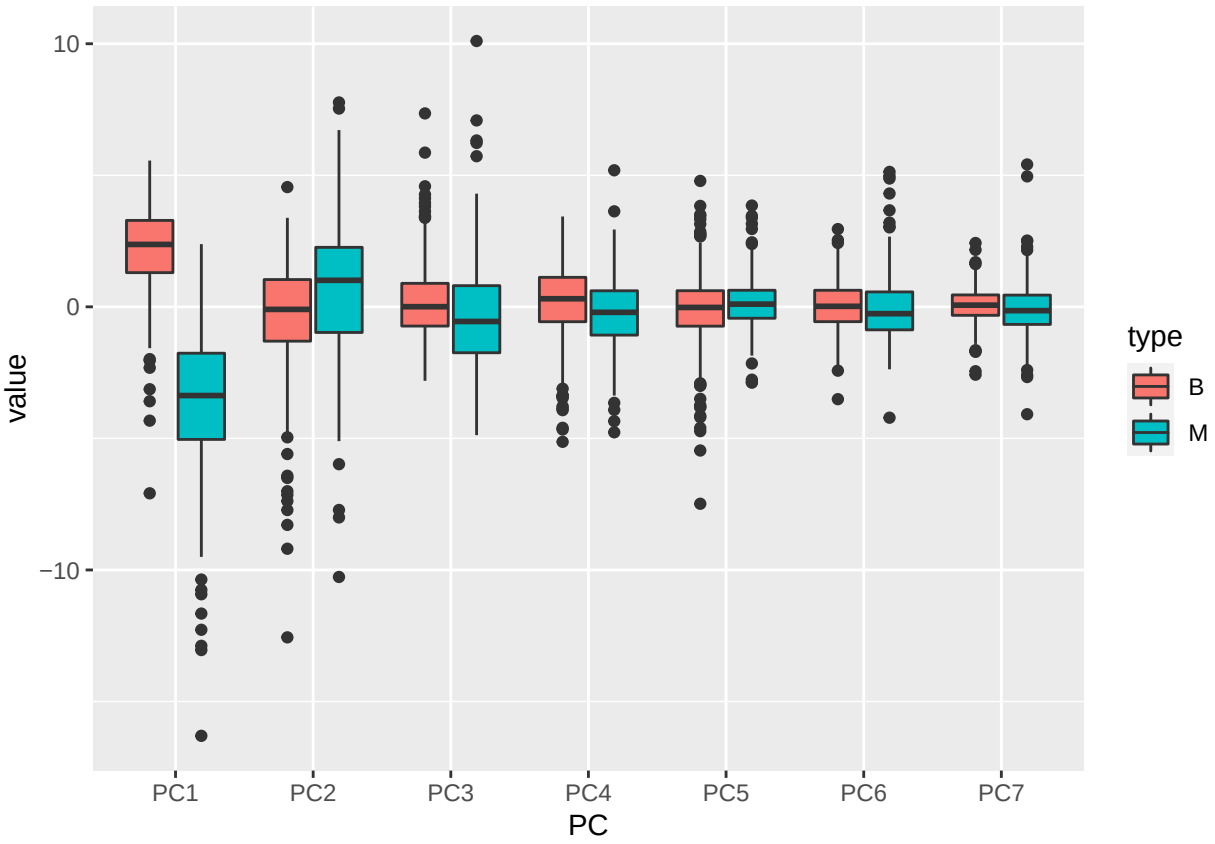
```r
data.frame(pca$x[,1:2], type = dat$y) %>%
  ggplot(aes(PC1, PC2, color = type)) +
  geom_point(alpha = 0.5)
```



Here we see that benign tumors tend to have higher values of PC1; we also observe that benign and malignant tumors have a similar spread of values for PC2. We also achieve much better delineation between classes than the previous analysis achieved. However, there are still some outliers present.o

Let's make box-plots of the first 7 principal components grouped by tumor type, because as we already discovered, the first 7 PC's explain 90% of the variance.

```
data.frame(type = dat$y, pca$x[,1:7]) %>%
  gather(key = "PC", value = "value", -type) %>%
  ggplot(aes(PC, value, fill = type)) +
  geom_boxplot()
```



Now that we have better idea of the predictive challenges ahead, let's make a new data frame that will hold the x_scaled features and the diagnosis (which will be stored as y).

```
x_scaled <- as.data.frame(x_scaled)
db <- x_scaled %>% cbind(y = dat$y)
```

The split

Time to split the data. The goal is to be able to predict as accurately as possible whether a new sample taken from a fine needle aspirate of a tumor is benign or malignant. Therefore, I want to split the data twice so that we can train and test algorithms on a subset before I run the final validation on a hold out set. Each split will be 20/80 (20% for testing and 80% for training).

```
# Make the Validation set.
set.seed(1654, sample.kind = "Rounding")
test_index <- createDataPartition(y = db$y, times = 1, p = 0.2, list = FALSE)
Val <- db[test_index,]
training <- db[-test_index,]

# take the training and split again into smaller test and train sets
set.seed(4123, sample.kind = "Rounding")
test_index2 <- createDataPartition(y = training$y, times = 1, p = 0.2, list = FALSE)
test <- training[test_index2, ]
train <- training[-test_index2,]

# remove the indexes and unnecessary objects
rm(test_index, test_index2, db, h, temp_dat, dat, x_scaled, x_centered, pca_sum,
```

```
   pca, d_features, d_samples, dist_MtoM, dist_MtoB, groups, var_explained)
```

Let's check the train and test sets for the prevalence of benign tumors.

```
mean(train$y=="B")
```

```
## [1] 0.6280992
```

```
mean(test$y=="B")
```

```
## [1] 0.6263736
```

Perfect, the prevalence is relatively the same among our train and test sets.

Models

I am now ready to move into generative models. I will compose several classification models, and then use an ensemble approach for the predictions on the test set. This method should improve the accuracy and minimize the confidence interval.

Logistic Regression

```
train_glm <- train(y~.,data = train, method = "glm")
glm_preds <- predict(train_glm, test)
glm_acc <- mean(glm_preds == test$y)
Accuracy_Results <- tibble(Method = "GLM", Accuracy = glm_acc)
Accuracy_Results %>% knitr::kable()
```

Method	Accuracy
GLM	0.9230769

Linear Discriminatory Analysis

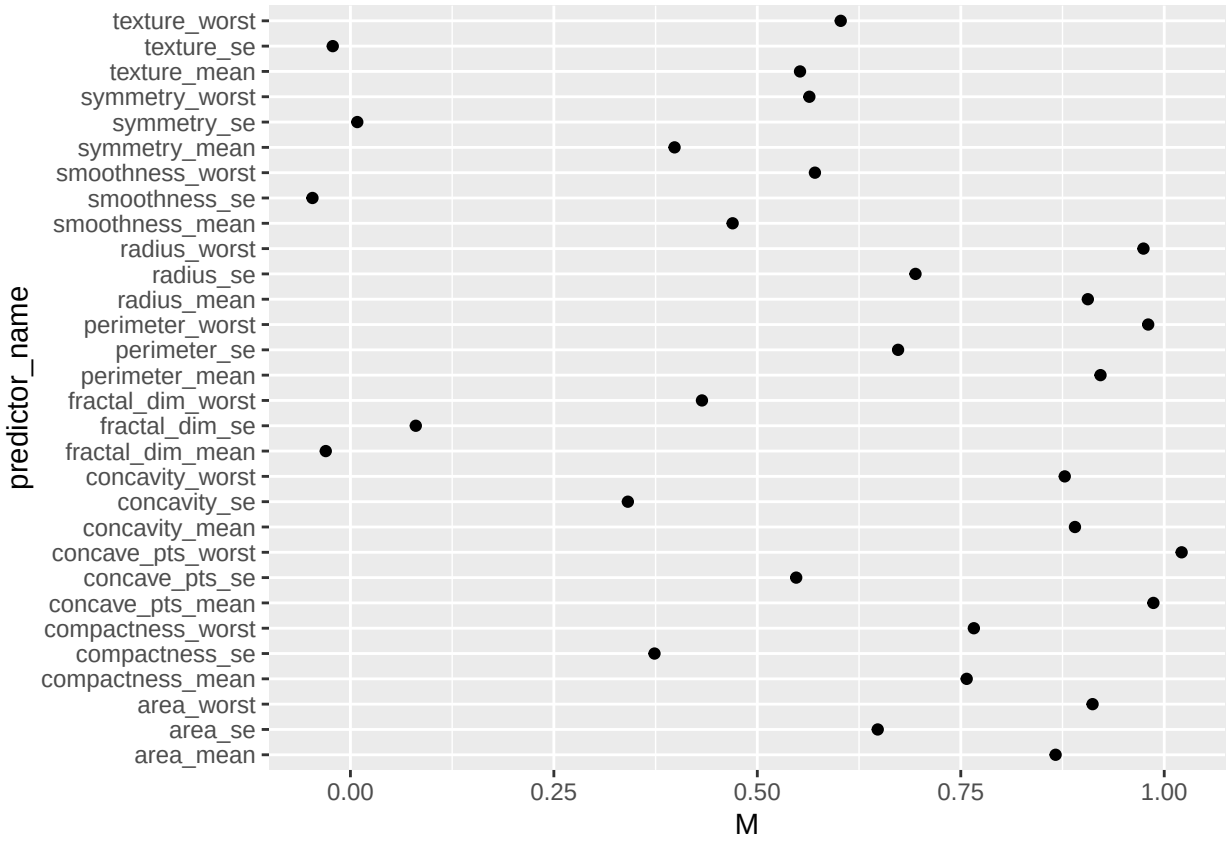
```
train_lda <- train(y~., data = train, method = "lda", preProcess = "center")
lda_preds <- predict(train_lda, test)
lda_acc <- mean(lda_preds == test$y)
Accuracy_Results <- bind_rows(Accuracy_Results,
                              tibble(Method = "LDA", Accuracy = lda_acc))
Accuracy_Results %>% knitr::kable()
```

Method	Accuracy
GLM	0.9230769
LDA	0.9560440

Let's examine the LDA model by making a plot of the predictors and their importance in the model for classifying malignant tumors.

```
t(train_lda$finalModel$means) %>%
  data.frame() %>%
  mutate(predictor_name = rownames(.)) %>%
  ggplot(aes(predictor_name, M, label=predictor_name)) +
  geom_point()+
```

```
  coord_flip()
```



Interesting, concave_pts_worst was the most important variable for classifying malignant tumors in the LDA model.

Quadratic Discriminant Analysis

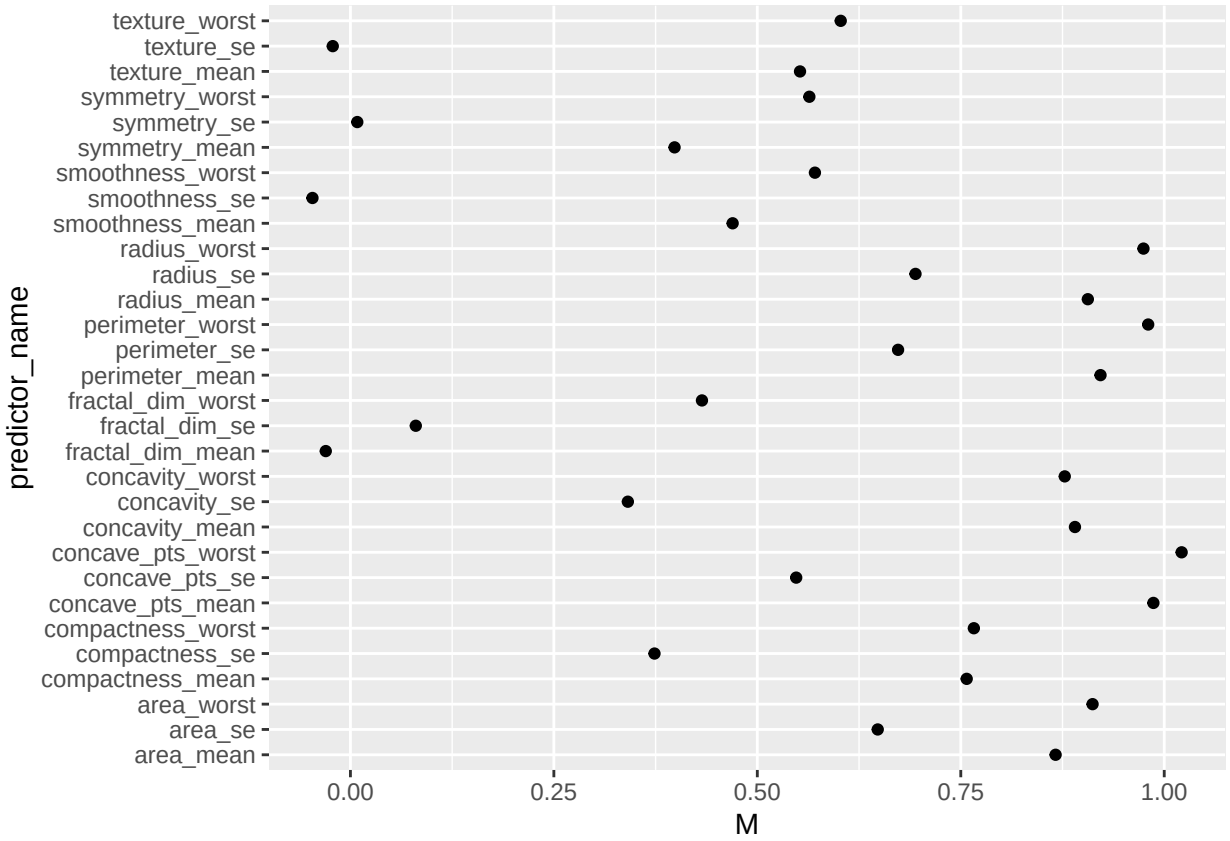
```
train_qda <- train(y~., train, method = "qda", preProcess = "center")
qda_preds <- predict(train_qda, test)
qda_acc <- mean(qda_preds == test$y)
Accuracy_Results <- bind_rows(Accuracy_Results,
                              tibble(Method = "QDA", Accuracy = qda_acc))
Accuracy_Results %>% knitr::kable()
```

Method	Accuracy
GLM	0.9230769
LDA	0.9560440
QDA	0.9670330

Let's again make a plot of the predictors and their importance to the QDA model in classifying malignant tumors.

```
t(train_qda$finalModel$means) %>% data.frame() %>%
  mutate(predictor_name = rownames(.)) %>%
  ggplot(aes(predictor_name, M, label=predictor_name)) +
  geom_point()+
```

```
  coord_flip()
```



So, we see the level of feature importance for the QDA model is the same as the LDA model.

GamLOESS

```
set.seed(5, sample.kind = "Rounding")
train_loess <- train(y~., data = train, method = "gamLoess")
loess_preds <- predict(train_loess, test)
loess_acc <- mean(loess_preds == test$y)
Accuracy_Results <- bind_rows(Accuracy_Results,
                              tibble(Method = "gamLoess", Accuracy = loess_acc))
Accuracy_Results %>% knitr::kable()
```

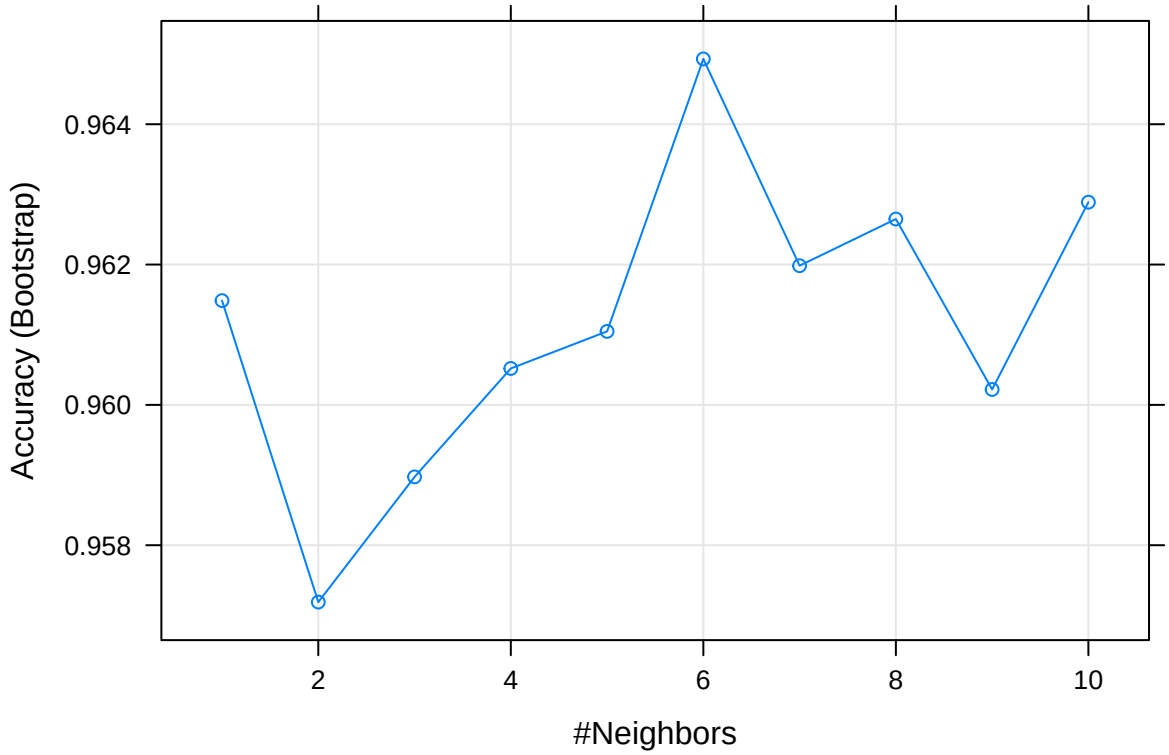
Method	Accuracy
GLM	0.9230769
LDA	0.9560440
QDA	0.9670330
gamLoess	0.9340659

K nearest neighbors

```
set.seed(7, sample.kind="Rounding")
tuning <- data.frame(k = seq(1, 10, 1))
train_knn <- train(y~.,
```

```
                    data = train,
                    method = "knn",
                    tuneGrid = tuning)

# show the best tune
plot(train_knn)
```




```
knn_preds <- predict(train_knn, test)
knn_acc <- mean(knn_preds == test$y)
Accuracy_Results <- bind_rows(Accuracy_Results,
                              tibble(Method = "KNN", Accuracy = knn_acc))
Accuracy_Results %>% knitr::kable()
```

Method	Accuracy
GLM	0.9230769
LDA	0.9560440
QDA	0.9670330
gamLoess	0.9340659
KNN	0.9340659

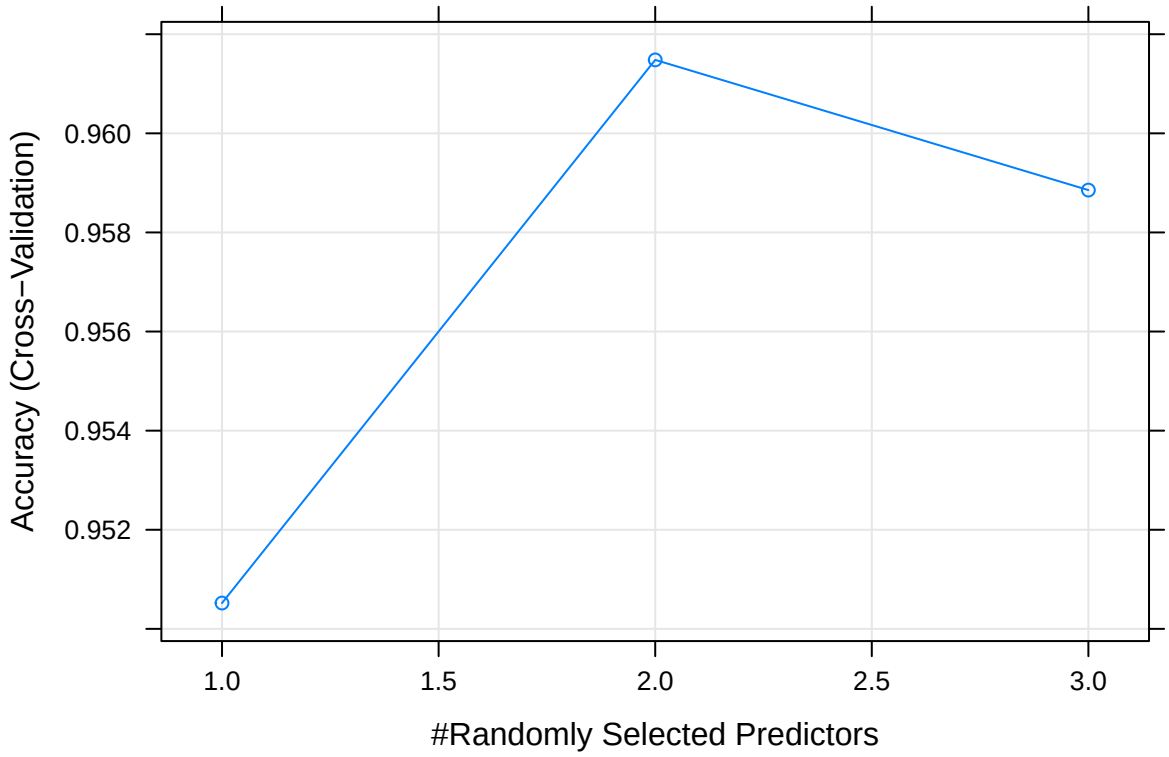
Random Forest

```
set.seed(9, sample.kind="Rounding")
tuning <- data.frame(mtry = c(1,2,3))
train_rf <- train(y~.,
                  data = train,
                  method = "rf",
                  tuneGrid = tuning,
```

```
                  trControl = trainControl(method = "cv",
                                           number = 10),
                  importance = TRUE)
```

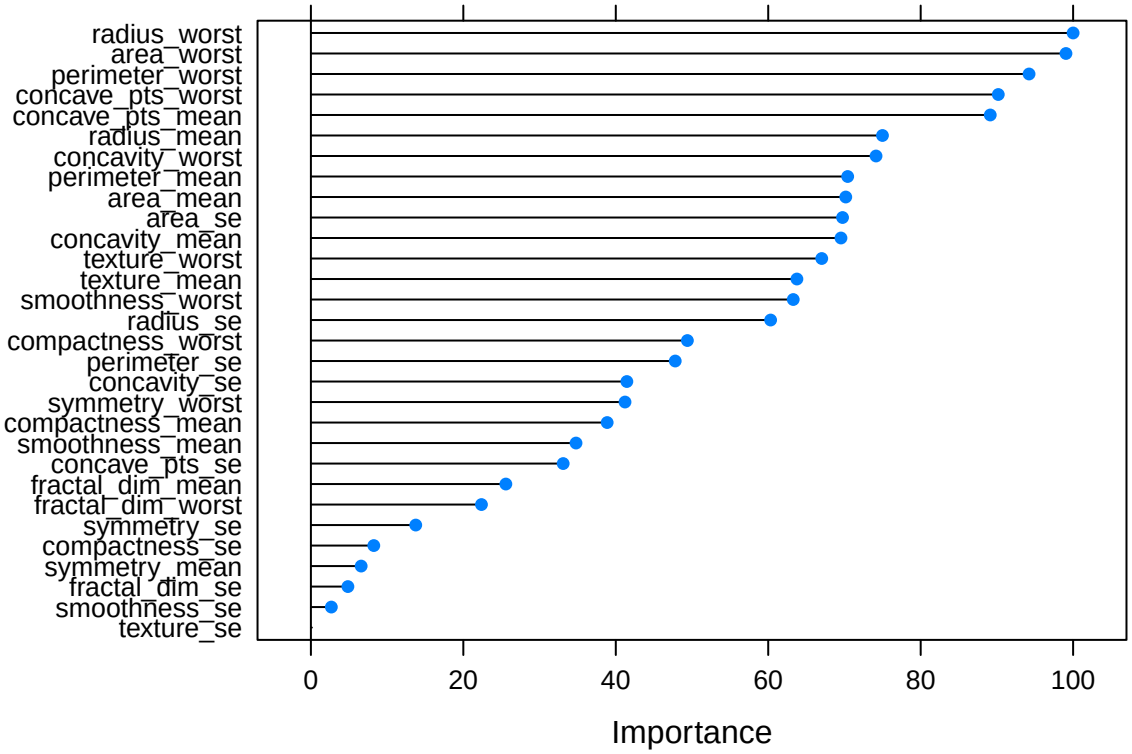
We can examine the tuning with a plot.

```
plot(train_rf)
```



Now that we know we have a reasonably tuned random forest model, let's examine the importance of the features to the classification process.

```
plot(varImp(train_rf))
```



OK, we find that many of the most important features are the "worst" features. We also find that the importance of features for the Random Forest model is different than what we found with the LDA and QDA models.

Time to run the predictions and report the accuracy.

```
rf_preds <- predict(train_rf, test)
rf_acc <- mean(rf_preds == test$y)
Accuracy_Results <- bind_rows(Accuracy_Results,
                              tibble(Method = "RF", Accuracy = rf_acc))
Accuracy_Results %>% knitr::kable()
```

Method	Accuracy
GLM	0.9230769
LDA	0.9560440
QDA	0.9670330
gamLoess	0.9340659
KNN	0.9340659
RF	0.9340659

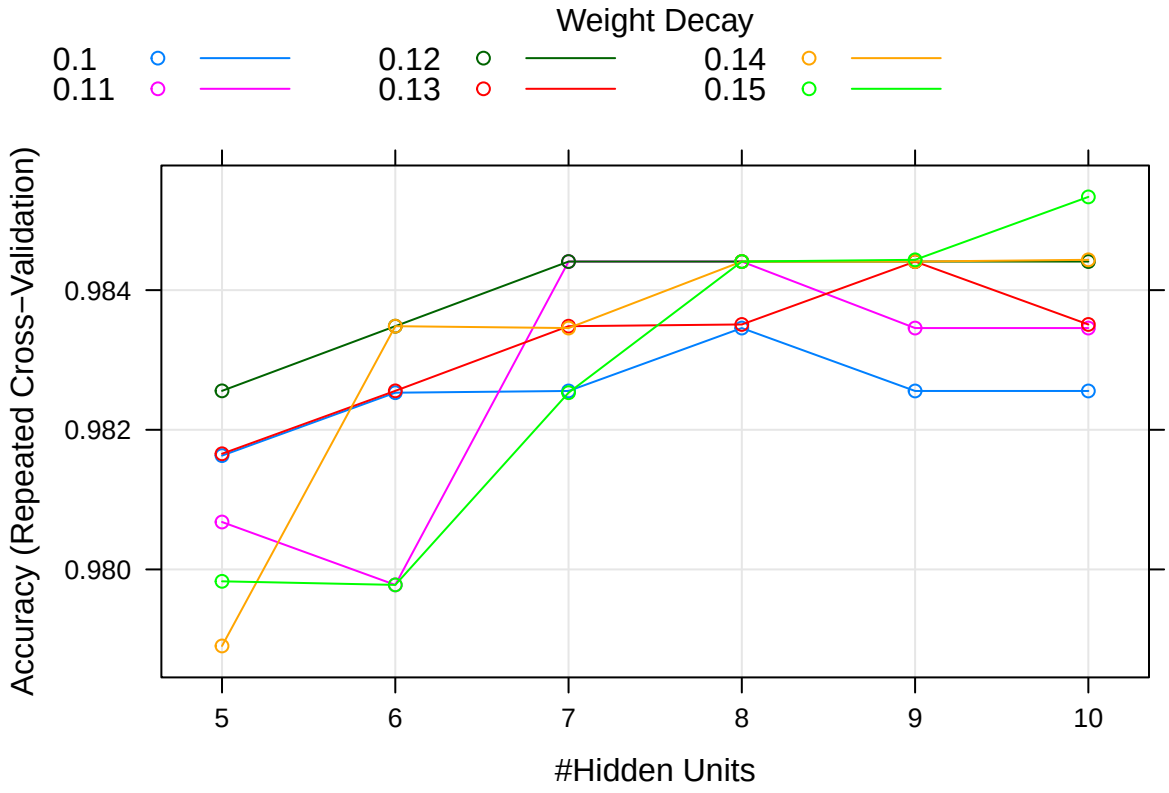
Neural Network with repeated Cross Validation

```
set.seed(2976, sample.kind = "Rounding")
# I will use cross validation in the train set in order to find the optimal
# hidden layers and decay.
tc <- trainControl(method = "repeatedcv", number = 10, repeats=3)
train_nn <- train(y~.,
                  data=train,
                  method='nnet',
```

```
                  linout=FALSE,
                  trace = FALSE,
                  trControl = tc,
                  tuneGrid=expand.grid(.size= seq(5,10,1),.decay=seq(0.1,0.15,0.01)))
```

Plot the different models to see the effect on accuracy

```
plot(train_nn)
```



Show the best Tune

```
train_nn$bestTune
```

```
##    size decay
## 36   10  0.15
```

```
nn_preds <- predict(train_nn, test)
nn_acc <- mean(nn_preds == test$y)
Accuracy_Results <- bind_rows(Accuracy_Results,
                              tibble(Method = "NNet", Accuracy = nn_acc))
Accuracy_Results %>% knitr::kable()
```

Method	Accuracy
GLM	0.9230769
LDA	0.9560440
QDA	0.9670330
gamLoess	0.9340659
KNN	0.9340659
RF	0.9340659
NNet	0.9560440

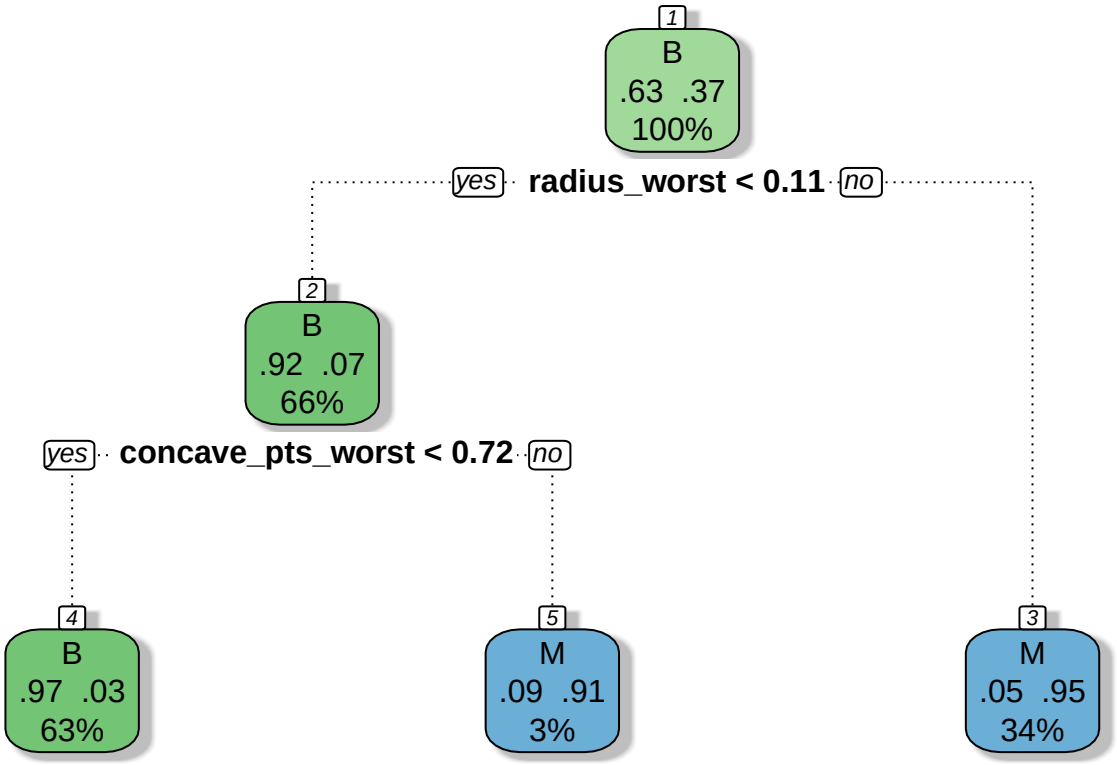
Recursive Partitioning - Rpart

```
set.seed(10, sample.kind = "Rounding")
train_rpart <- train(y ~ .,
                     data = train,
                     method = "rpart",
                     trControl = trainControl(method = "cv", number = 10),
                     tuneGrid = data.frame(cp = seq(.022, .023, .0001)))

# Show the best tune
train_rpart$bestTune
```

```
##       cp
## 11 0.023
```

```
# plot the resulting decision tree from the rpart model.
fancyRpartPlot(train_rpart$finalModel, yesno = 2)
```




```
rpart_preds <- predict(train_rpart, test)
rpart_acc <- mean(rpart_preds == test$y)
Accuracy_Results <- bind_rows(Accuracy_Results,
                              tibble(Method = "RPart", Accuracy = rpart_acc))
Accuracy_Results %>% knitr::kable()
```

Method	Accuracy
GLM	0.9230769
LDA	0.9560440
QDA	0.9670330
gamLoess	0.9340659

Method	Accuracy
KNN	0.9340659
RF	0.9340659
NNet	0.9560440
RPart	0.9010989

All of the models we have run so far have an accuracy over 90%. We know from examining the feature importance in the models that they are weighting the features differently for their use in classifying the tumor samples. Therefore, we should expect an increase in sensitivity by creating an ensemble.

First, let's examine the most accurate model so far with a confusion matrix.

```
confusionMatrix(as.factor(qda_preds), as.factor(test$y))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 56  2
##          M  1 32
##
##                Accuracy : 0.967
##                  95% CI : (0.9067, 0.9931)
##     No Information Rate : 0.6264
##     P-Value [Acc > NIR] : 8.87e-15
##
##                   Kappa : 0.9291
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9825
##             Specificity : 0.9412
##          Pos Pred Value : 0.9655
##          Neg Pred Value : 0.9697
##              Prevalence : 0.6264
##          Detection Rate : 0.6154
##    Detection Prevalence : 0.6374
##       Balanced Accuracy : 0.9618
##
##        'Positive' Class : B
##
```

We will compare these results to a confusion matrix of an ensemble approach.

Ensemble Method

I will create a new data frame that holds all of the benign predictions from the previous generative models.

```
ensemble <- cbind(glm = glm_preds == "B", lda = lda_preds == "B",
                  qda = qda_preds == "B", loess = loess_preds == "B",
                  rf = rf_preds == "B", nn = nn_preds == "B",
                  rp = rpart_preds == "B", knn = knn_preds == "B")
```

I will say that if more than half of the algorithms predicted benign then the ensemble will predict benign. This also means that if less than half predict benign then the ensemble will predict malignant.

```
ensemble_preds <- ifelse(rowMeans(ensemble) > 0.5, "B", "M")
```

Let's check the accuracy of the ensemble method against the test set and present the findings by attaching it to our running table.

```
ensemble_acc <- mean(ensemble_preds == test$y)
Accuracy_Results <- bind_rows(Accuracy_Results,
                              tibble(Method = "Ensemble",
                                     Accuracy = ensemble_acc))
Accuracy_Results %>% knitr::kable()
```

Method	Accuracy
GLM	0.9230769
LDA	0.9560440
QDA	0.9670330
gamLoess	0.9340659
KNN	0.9340659
RF	0.9340659
NNet	0.9560440
RPart	0.9010989
Ensemble	0.9670330

Now, let's examine a confusion matrix to see if we have better results than the QDA model had on it's own.

```
confusionMatrix(as.factor(ensemble_preds), as.factor(test$y))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 57  3
##          M  0 31
##
##                Accuracy : 0.967
##                  95% CI : (0.9067, 0.9931)
##     No Information Rate : 0.6264
##     P-Value [Acc > NIR] : 8.87e-15
##
##                   Kappa : 0.9283
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.9118
##          Pos Pred Value : 0.9500
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6264
##          Detection Rate : 0.6264
##    Detection Prevalence : 0.6593
##       Balanced Accuracy : 0.9559
##
##        'Positive' Class : B
##
```

OK! The ensemble method has improved the sensitivity. We also see that the negative prediction value is 1. This means that every time (100% of the time) the ensemble method predicted malignancy, the tumor was indeed malignant.

Let's repeat the above steps with the data from the first split, and run the final validation on the hold out set.

Validation

```
set.seed(1, sample.kind = "Rounding")
train_glm <- train(y~.,data = training, method = "glm")

set.seed(2, sample.kind = "Rounding")
train_lda <- train(y~., data = training, method = "lda")

set.seed(3, sample.kind = "Rounding")
train_qda <- train(y~., training, method = "qda")

set.seed(4, sample.kind = "Rounding")
train_loess <- train(y~., data = training, method = "gamLoess")

set.seed(5, sample.kind="Rounding")
tuning <- data.frame(k = seq(3, 21, 2))
train_knn <- train(y~.,
                   data = training,
                   method = "knn",
                   tuneGrid = tuning)

set.seed(6, sample.kind="Rounding")
tuning <- data.frame(mtry = c(1,2,3))
train_rf <- train(y~.,
                  data = training,
                  method = "rf",
                  tuneGrid = tuning,
                  trControl = trainControl(method = "cv",
                                           number = 10),
                  importance = TRUE)


set.seed(7, sample.kind = "Rounding")
# I will use cross validation in the train set in order to find the optimal
# hidden layers and decay.
tc <- trainControl(method = "repeatedcv", number = 10, repeats=3)
train_nn <- train(y~.,
                  data=training,
                  method='nnet',
                  linout=FALSE,
                  trace = FALSE,
                  trControl = tc,
                  tuneGrid=expand.grid(.size= seq(5,10,1),
                                       .decay=seq(0.15,0.2,0.01)))

set.seed(8, sample.kind = "Rounding")
train_rpart <- train(y ~ .,
```

```
                      data = training,
                      method = "rpart",
                      trControl = trainControl(method = "cv", number = 10),
                      tuneGrid = data.frame(cp = seq(.022, .023, .0001)))


val_preds_glm <- predict(train_glm, Val)
val_preds_lda <- predict(train_lda, Val)
val_preds_qda <- predict(train_qda, Val)
val_preds_loess <- predict(train_loess, Val)
val_preds_rf <- predict(train_rf, Val)
val_preds_nn <- predict(train_nn, Val)
val_preds_rp <- predict(train_rpart, Val)
val_preds_knn <- predict(train_knn, Val)

Val_Ensemble <- cbind(glm = val_preds_glm == "B", lda = val_preds_lda == "B",
                      qda = val_preds_qda == "B", loess = val_preds_loess == "B",
                      rf = val_preds_rf == "B", nn = val_preds_nn == "B",
                      rp = val_preds_rp == "B", knn = val_preds_knn == "B")

Val_Ensemble_preds <- ifelse(rowMeans(Val_Ensemble)>0.5, "B", "M")
Val_Acc <- mean(Val_Ensemble_preds == Val$y)

Accuracy_Results <- bind_rows(Accuracy_Results,
                              tibble(Method = "Ensemble Validation",
                                     Accuracy = Val_Acc))
Accuracy_Results %>% knitr::kable()
```

Method	Accuracy
GLM	0.9230769
LDA	0.9560440
QDA	0.9670330
gamLoess	0.9340659
KNN	0.9340659
RF	0.9340659
NNet	0.9560440
RPart	0.9010989
Ensemble	0.9670330
Ensemble Validation	0.9913043

```
confusionMatrix(as.factor(Val_Ensemble_preds), as.factor(Val$y))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 72  1
##          M  0 42
##
##                Accuracy : 0.9913
##                  95% CI : (0.9525, 0.9998)
##     No Information Rate : 0.6261
```

```
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9813
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9767
##          Pos Pred Value : 0.9863
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6261
##          Detection Rate : 0.6261
##    Detection Prevalence : 0.6348
##       Balanced Accuracy : 0.9884
##
##        'Positive' Class : B
##
```

Summary

The ensemble approach has yielded an accuracy above 99%, with a 95% confidence interval between 95.25% and 99.98%, and an overall balanced accuracy of 98.84%. The addition of new data for training also provided an improvement in the positive prediction value (benign tumors). Just like the first split, the ensemble method proves 100% accurate for predicting malignant tumors on the final validation hold out.

> "In 2020, there were 2.3 million women diagnosed with breast cancer and 685,000 deaths globally. As of the end of 2020, there were 7.8 million women alive who were diagnosed with breast cancer in the past 5 years, making it the world's most prevalent cancer. There are more lost disability-adjusted life years (DALYs) by women to breast cancer globally than any other type of cancer. Breast cancer occurs in every country of the world in women at any age after puberty but with increasing rates in later life."
>
> — World Health Organization

Hopefully, with the aid of machine learning, physicians can more speedily and accurately identify malignant growths via fine needle aspiration.

I hope you have enjoyed reading this project.