Using a Machine Learning Ensemble to Make Breast Cancer Predictions

Jarred Priester

1/11/2022

1. Overview

Something that we would all like to eradicate from our society is cancer. Unfortunately, cancer has effected our lives far too often. Thankfully, cancer research has advanced quite a lot in the past decades thanks in large part to advances in technology and in particular, machine learning. Hopefully this project will shine some light on the frame work of how machine learning can be used to further cancer research.

1.1 Decription of dataset

The Breast Cancer Wisconsin (Diagnostic) Data Set is a popular data set from the University of California Irvine Machine Learning Repository. The data set consist of 529 rows and 32 columns. Each row represents a tumor sample and each column represents a feature, more details are below.

The following is from UCI's Machine Learning Repository website:

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.

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

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)

1.2 Goal of the project

The goal of this project will be to successfully create a model that could classify the given tumor samples into factors of malignant or benign. The metric we will use to determine success is the F1 score. The goal is to create a model that can achieve a F1 score of .9 or higher.

1.3 Steps to achieve this goal

To achieve this goal we will first download, clean and analyze the dataset. We will then run 5 different algorithms that come up with the binary classification predictions of malignant or benign. We will then combine them to create an ensemble that takes the classification that appears the most for each sample . Lastly we will create a table of results and find the model with the highest F1 score.

Data Cleaning

2.1 downloading the data

```
  #installing required packages if not previouly installed
if(!require(matrixStats)) install.packages("matrixStats")
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(caret)) install.packages("caret")
if(!require(dslabs)) install.packages("dslabs")
if(!require(dplyr)) install.packages("dplyr")
if(!require(tidyr)) install.packages("tidyr")
if(!require(ggthemes)) install.packages("ggthemes")
if(!require(knitr)) install.packages("knitr")

#setting digits to 3 places
options(digits = 3)

#downloading the libraries
library(matrixStats)
library(tidyverse)
library(caret)
library(dslabs)
library(dplyr)
library(tidyr)
library(ggthemes)
library(knitr)

#downloading the data from the dslabs library
data(brca)
```

the data are in two list. Let's take a look at the dimensions of both list

```
dim(brca$x)
```

```
## [1] 569  30
```

```
dim(brca$y)
```

```
## NULL
```

```r
head(brca$y)
```

```
## [1] B B B B B B
## Levels: B M
```

taking a look at the brca$x data

```r
head(brca$x)
```

```
##      radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## [1,]        13.5         14.4           87.5       566          0.0978
## [2,]        13.1         15.7           85.6       520          0.1075
## [3,]         9.5         12.4           60.3       274          0.1024
## [4,]        13.0         18.4           82.6       524          0.0898
## [5,]         8.2         16.8           51.7       202          0.0860
## [6,]        12.1         14.6           78.0       449          0.1031
##      compactness_mean concavity_mean concave_pts_mean symmetry_mean
## [1,]           0.0813         0.0666          0.04781         0.188
## [2,]           0.1270         0.0457          0.03110         0.197
## [3,]           0.0649         0.0296          0.02076         0.181
## [4,]           0.0377         0.0256          0.02923         0.147
## [5,]           0.0594         0.0159          0.00592         0.177
## [6,]           0.0909         0.0659          0.02749         0.168
##      fractal_dim_mean radius_se texture_se perimeter_se area_se smoothness_se
## [1,]           0.0577     0.270      0.789         2.06   23.56       0.00846
## [2,]           0.0681     0.185      0.748         1.38   14.67       0.00410
## [3,]           0.0690     0.277      0.977         1.91   15.70       0.00961
## [4,]           0.0586     0.184      2.342         1.17   14.16       0.00435
## [5,]           0.0650     0.156      0.957         1.09    8.21       0.00897
## [6,]           0.0604     0.264      0.729         1.85   19.87       0.00549
##      compactness_se concavity_se concave_pts_se symmetry_se fractal_dim_se
## [1,]         0.0146       0.0239        0.01315      0.0198        0.00230
## [2,]         0.0190       0.0170        0.00649      0.0168        0.00243
## [3,]         0.0143       0.0198        0.01421      0.0203        0.00297
## [4,]         0.0049       0.0134        0.01164      0.0267        0.00178
## [5,]         0.0165       0.0159        0.00592      0.0257        0.00258
## [6,]         0.0143       0.0232        0.00566      0.0143        0.00242
##      radius_worst texture_worst perimeter_worst area_worst smoothness_worst
## [1,]        15.11          19.3            99.7        711            0.144
## [2,]        14.50          20.5            96.1        630            0.131
## [3,]        10.23          15.7            65.1        315            0.132
## [4,]        13.30          22.8            84.5        546            0.097
## [5,]         8.96          22.0            57.3        242            0.130
## [6,]        13.76          20.7            89.9        583            0.149
##      compactness_worst concavity_worst concave_pts_worst symmetry_worst
## [1,]            0.1773          0.2390            0.1288          0.298
## [2,]            0.2776          0.1890            0.0728          0.318
## [3,]            0.1148          0.0887            0.0623          0.245
## [4,]            0.0462          0.0483            0.0501          0.199
## [5,]            0.1357          0.0688            0.0256          0.310
## [6,]            0.2156          0.3050            0.0655          0.275
##      fractal_dim_worst
```

```
## [1,]            0.0726
## [2,]            0.0818
## [3,]            0.0777
## [4,]            0.0617
## [5,]            0.0741
## [6,]            0.0830
```

changing brca$x to just x

```
x <- brca$x
```

changing brca$y to just y

```
y <- brca$y
```

taking a look at the variables in x

```
colnames(x)
```

```
##  [1] "radius_mean"       "texture_mean"      "perimeter_mean"
##  [4] "area_mean"         "smoothness_mean"   "compactness_mean"
##  [7] "concavity_mean"    "concave_pts_mean"  "symmetry_mean"
## [10] "fractal_dim_mean"  "radius_se"         "texture_se"
## [13] "perimeter_se"      "area_se"           "smoothness_se"
## [16] "compactness_se"    "concavity_se"      "concave_pts_se"
## [19] "symmetry_se"       "fractal_dim_se"    "radius_worst"
## [22] "texture_worst"     "perimeter_worst"   "area_worst"
## [25] "smoothness_worst"  "compactness_worst" "concavity_worst"
## [28] "concave_pts_worst" "symmetry_worst"    "fractal_dim_worst"
```

structure of x

```
str(x)
```

```
##  num [1:569, 1:30] 13.5 13.1 9.5 13 8.2 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:30] "radius_mean" "texture_mean" "perimeter_mean" "area_mean" ...
```

summary of x

```
summary(x)
```

```
##   radius_mean     texture_mean  perimeter_mean    area_mean    smoothness_mean
##  Min.   : 6.98   Min.   : 9.7   Min.   : 43.8   Min.   : 144   Min.   :0.0526
##  1st Qu.:11.70   1st Qu.:16.2   1st Qu.: 75.2   1st Qu.: 420   1st Qu.:0.0864
##  Median :13.37   Median :18.8   Median : 86.2   Median : 551   Median :0.0959
##  Mean   :14.13   Mean   :19.3   Mean   : 92.0   Mean   : 655   Mean   :0.0964
##  3rd Qu.:15.78   3rd Qu.:21.8   3rd Qu.:104.1   3rd Qu.: 783   3rd Qu.:0.1053
##  Max.   :28.11   Max.   :39.3   Max.   :188.5   Max.   :2501   Max.   :0.1634
##  compactness_mean concavity_mean  concave_pts_mean symmetry_mean
```

```
##  Min.   :0.019    Min.   :0.000   Min.   :0.0000   Min.   :0.106
##  1st Qu.:0.065    1st Qu.:0.030   1st Qu.:0.0203   1st Qu.:0.162
##  Median :0.093    Median :0.062   Median :0.0335   Median :0.179
##  Mean   :0.104    Mean   :0.089   Mean   :0.0489   Mean   :0.181
##  3rd Qu.:0.130    3rd Qu.:0.131   3rd Qu.:0.0740   3rd Qu.:0.196
##  Max.   :0.345    Max.   :0.427   Max.   :0.2012   Max.   :0.304
##  fractal_dim_mean   radius_se       texture_se    perimeter_se      area_se
##  Min.   :0.0500   Min.   :0.112   Min.   :0.36   Min.   : 0.76   Min.   :  7
##  1st Qu.:0.0577   1st Qu.:0.232   1st Qu.:0.83   1st Qu.: 1.61   1st Qu.: 18
##  Median :0.0615   Median :0.324   Median :1.11   Median : 2.29   Median : 25
##  Mean   :0.0628   Mean   :0.405   Mean   :1.22   Mean   : 2.87   Mean   : 40
##  3rd Qu.:0.0661   3rd Qu.:0.479   3rd Qu.:1.47   3rd Qu.: 3.36   3rd Qu.: 45
##  Max.   :0.0974   Max.   :2.873   Max.   :4.88   Max.   :21.98   Max.   :542
##  smoothness_se     compactness_se    concavity_se   concave_pts_se
##  Min.   :0.00171   Min.   :0.0023   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.00517   1st Qu.:0.0131   1st Qu.:0.015   1st Qu.:0.0076
##  Median :0.00638   Median :0.0204   Median :0.026   Median :0.0109
##  Mean   :0.00704   Mean   :0.0255   Mean   :0.032   Mean   :0.0118
##  3rd Qu.:0.00815   3rd Qu.:0.0324   3rd Qu.:0.042   3rd Qu.:0.0147
##  Max.   :0.03113   Max.   :0.1354   Max.   :0.396   Max.   :0.0528
##   symmetry_se     fractal_dim_se     radius_worst  texture_worst
##  Min.   :0.0079   Min.   :0.00089   Min.   : 7.9   Min.   :12.0
##  1st Qu.:0.0152   1st Qu.:0.00225   1st Qu.:13.0   1st Qu.:21.1
##  Median :0.0187   Median :0.00319   Median :15.0   Median :25.4
##  Mean   :0.0205   Mean   :0.00379   Mean   :16.3   Mean   :25.7
##  3rd Qu.:0.0235   3rd Qu.:0.00456   3rd Qu.:18.8   3rd Qu.:29.7
##  Max.   :0.0790   Max.   :0.02984   Max.   :36.0   Max.   :49.5
##  perimeter_worst   area_worst   smoothness_worst compactness_worst
##  Min.   : 50.4   Min.   : 185   Min.   :0.0712   Min.   :0.027
##  1st Qu.: 84.1   1st Qu.: 515   1st Qu.:0.1166   1st Qu.:0.147
##  Median : 97.7   Median : 686   Median :0.1313   Median :0.212
##  Mean   :107.3   Mean   : 881   Mean   :0.1324   Mean   :0.254
##  3rd Qu.:125.4   3rd Qu.:1084   3rd Qu.:0.1460   3rd Qu.:0.339
##  Max.   :251.2   Max.   :4254   Max.   :0.2226   Max.   :1.058
##  concavity_worst concave_pts_worst symmetry_worst  fractal_dim_worst
##  Min.   :0.000   Min.   :0.0000    Min.   :0.156   Min.   :0.0550
##  1st Qu.:0.114   1st Qu.:0.0649    1st Qu.:0.250   1st Qu.:0.0715
##  Median :0.227   Median :0.0999    Median :0.282   Median :0.0800
##  Mean   :0.272   Mean   :0.1146    Mean   :0.290   Mean   :0.0839
##  3rd Qu.:0.383   3rd Qu.:0.1614    3rd Qu.:0.318   3rd Qu.:0.0921
##  Max.   :1.252   Max.   :0.2910    Max.   :0.664   Max.   :0.2075
```

2.2 missing information

taking a look to see if there are any NAs or blank cells

```
colSums(is.na(x))
```

```
##       radius_mean      texture_mean    perimeter_mean         area_mean
##                 0                 0                 0                 0
##   smoothness_mean  compactness_mean    concavity_mean  concave_pts_mean
##                 0                 0                 0                 0
```

```
##     symmetry_mean  fractal_dim_mean         radius_se        texture_se
##                 0                 0                 0                 0
##      perimeter_se           area_se     smoothness_se    compactness_se
##                 0                 0                 0                 0
##      concavity_se    concave_pts_se       symmetry_se    fractal_dim_se
##                 0                 0                 0                 0
##      radius_worst     texture_worst   perimeter_worst        area_worst
##                 0                 0                 0                 0
##  smoothness_worst compactness_worst   concavity_worst concave_pts_worst
##                 0                 0                 0                 0
##    symmetry_worst fractal_dim_worst
##                 0                 0
```

```
sum(x == "")
```

```
## [1] 0
```

There is no missing information so we now move on to the next step.

2.3 scaling the matrix

After looking at the summary of x we can see that the features do not have the same ranges. In fact some are quite larger than others. So to avoid any features influencing the models in an adverse way, we are now going to scale the matrix

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

checking the first column's standard deviation, should be close to 1 since we scaled the matrix

```
sd(x_scaled[,1])
```

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

checking the first column's median value, should be close to 0 after scaling

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

```
## [1] -0.215
```

3. Exploratory Data Analysis

3.1 exploring the data

Is our outcomes balanced?

our outcomes are not balance around 2/3 are benign (not cancerous)

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

```
## [1] 0.373
```

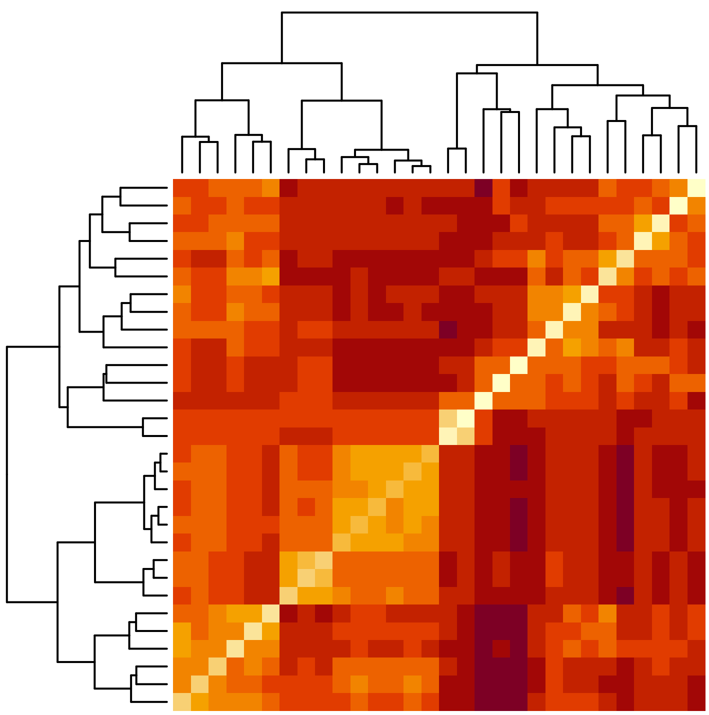
```r
mean(y == "B")
```

```
## [1] 0.627
```

3.2 Visialization

Next we will create a Heatmap in order to get a visual at how the features correlate to each other, if at all.

```r
d_features <- dist(t(x_scaled))
heatmap(as.matrix(d_features), labRow = NA, labCol = NA)
```



We can see that there is correlation throughout the data set so there is some promise that we will be able to classify the data accurately.

Hierarchical clustering

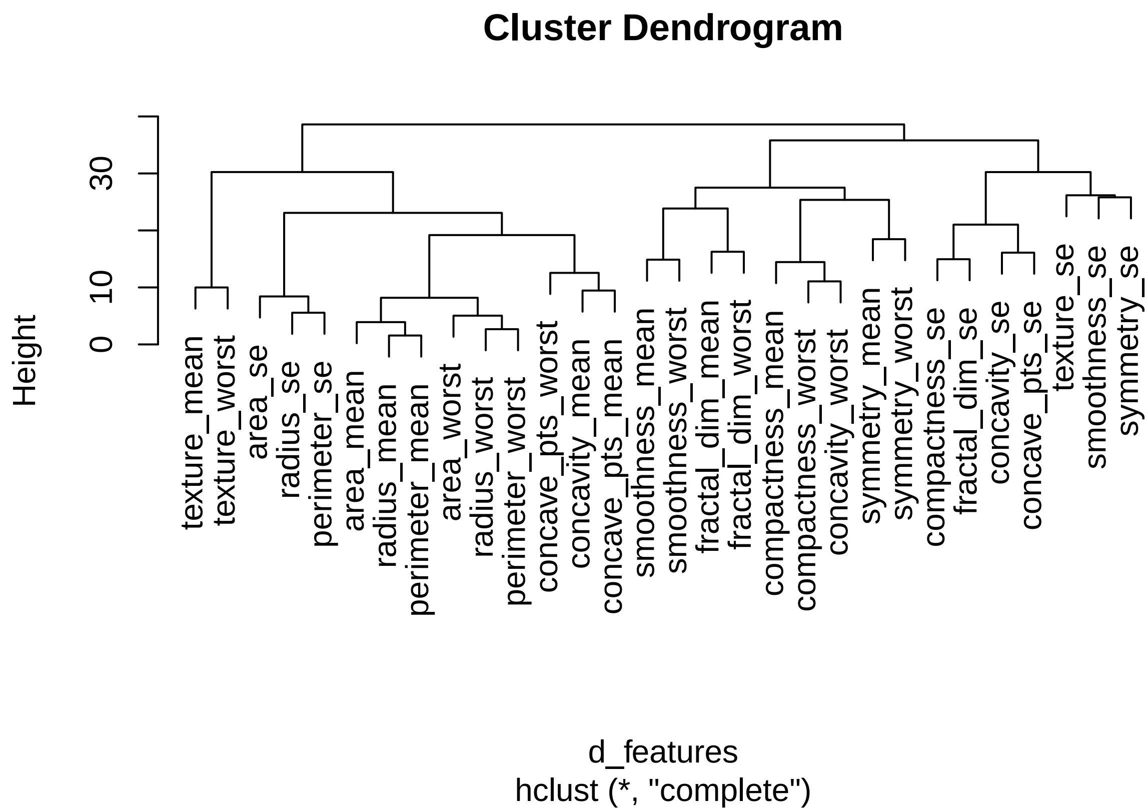
```r
h <- hclust(d_features)
groups <- cutree(h, k = 5)
groups
```

```
##       radius_mean      texture_mean    perimeter_mean         area_mean
##                 1                 2                 1                 1
##   smoothness_mean  compactness_mean    concavity_mean  concave_pts_mean
##                 3                 3                 1                 1
##     symmetry_mean  fractal_dim_mean         radius_se        texture_se
##                 3                 3                 1                 4
##      perimeter_se           area_se     smoothness_se    compactness_se
##                 1                 1                 4                 5
##      concavity_se    concave_pts_se       symmetry_se    fractal_dim_se
##                 5                 5                 4                 5
##      radius_worst     texture_worst   perimeter_worst        area_worst
##                 1                 2                 1                 1
##  smoothness_worst compactness_worst   concavity_worst concave_pts_worst
##                 3                 3                 3                 1
##    symmetry_worst fractal_dim_worst
##                 3                 3
```

```
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"
```

```
plot(h)
```

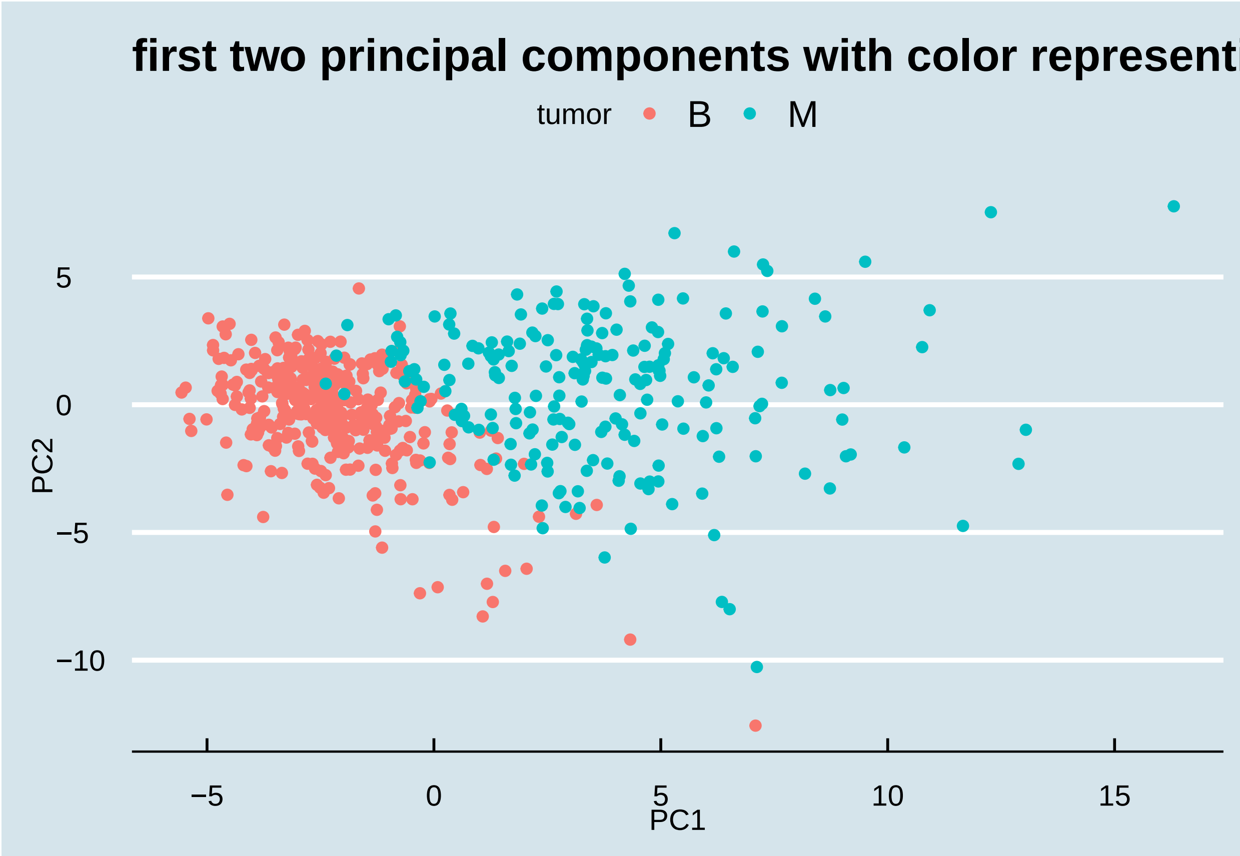


PCA: proportion of variance

```
pc <- prcomp(x_scaled)
```

Plot of the first two principal components with color representing tumor type

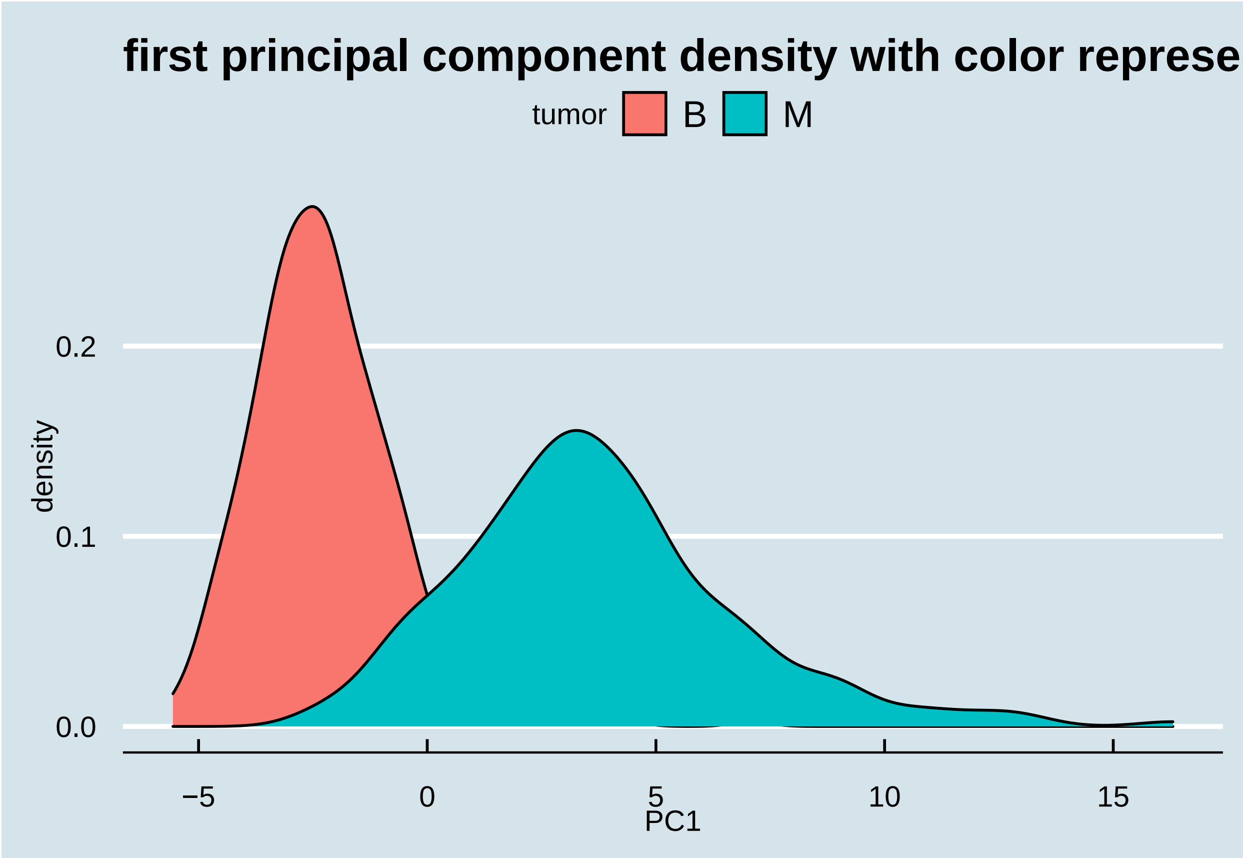
```
#(benign/malignant)
data.frame(pc$x[,1:2], tumor=brca$y) %>%
  ggplot(aes(PC1,PC2, fill = tumor, color = tumor))+
  geom_point() +
  labs(title = "first two principal components with color representing tumor type") +
  theme_economist()
```



We can see a clear separation between the first two components by tumor type. This tells us that we should be able to classify this data into malignant and benign with high accuracy.

plot showing the density of first principal component

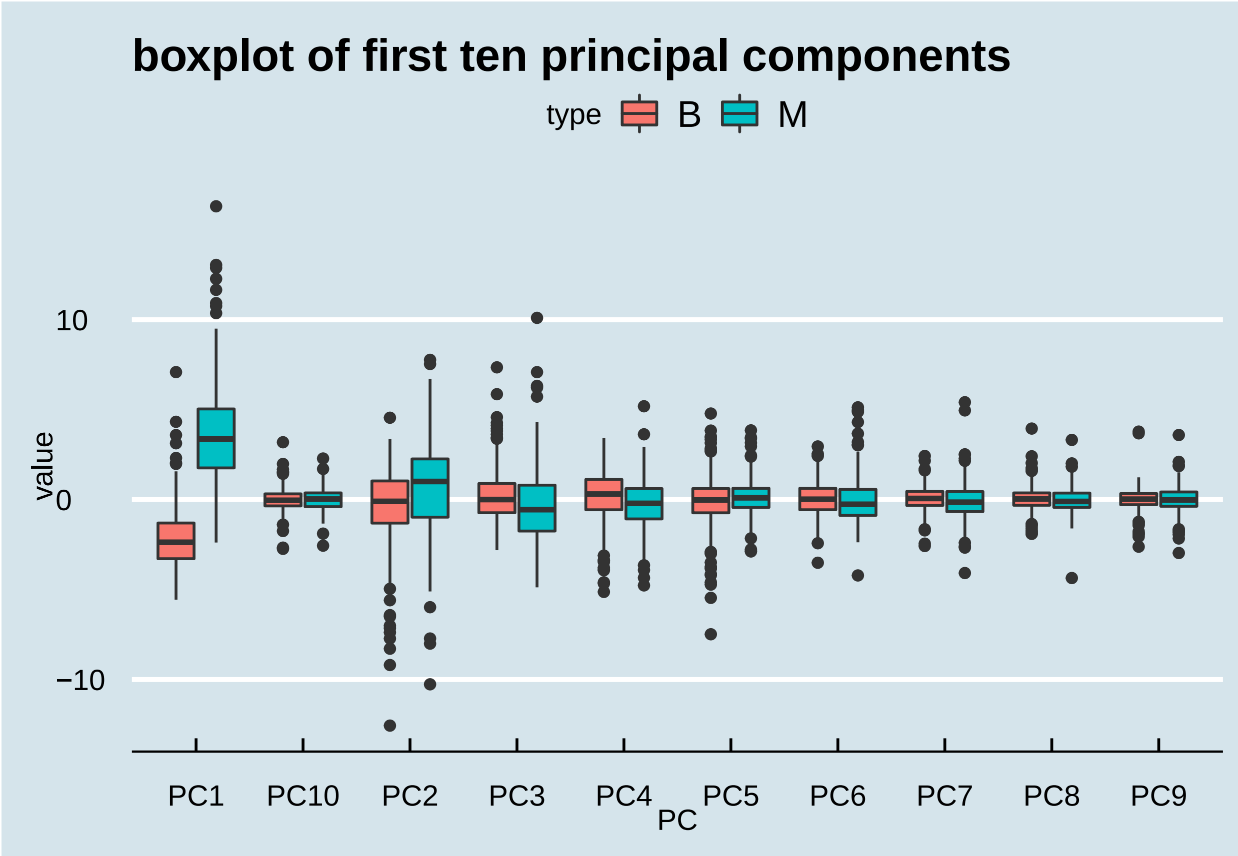
```
data.frame(pc$x[,1:2], tumor=brca$y) %>%
  ggplot(aes(PC1,fill = tumor))+
  geom_density() +
  labs(title = "first principal component density with color representing tumor type") +
  theme_economist()
```



Same information as the scatter plot but this time as a density plot. Again, you can see the separate between the two tumor types

boxplot of first ten principal components

```
data.frame(type = brca$y, pc$x[,1:10]) %>%
  gather(key = "PC", value = "value", -type) %>%
  ggplot(aes(PC, value, fill = type)) +
  geom_boxplot() +
  ggtitle("boxplot of first ten principal components") +
  theme_economist()
```



Here we can see that the malignant and benign interquartiles do not overlap, meaning there is separation in the data. That will help the models be able to classify the data.

4 Models

4.1 setting up the models

Creating the training and test sets

```
# set.seed(1) if using R 3.5 or earlier
set.seed(30, sample.kind = "Rounding")    # if using R 3.6 or later
test_index <- createDataPartition(brca$y, times = 1, p = 0.2, list = FALSE)
test_x <- x_scaled[test_index,]
test_y <- y[test_index]
train_x <- x_scaled[-test_index,]
train_y <- y[-test_index]
```

What proportion of the training set is benign?

```
mean(train_y == "B")
```

```
## [1] 0.628
```

What proportion of the test set is benign?

```
mean(test_y == "B")
```

```
## [1] 0.626
```

Will be using k-fold cross validation on all the algorithms creating the k-fold parameters, k is 10

```
set.seed(30, sample.kind = "Rounding")
control <- trainControl(method = "cv", number = 10, p = .9)
```

4.2 logistic regression

training the model using the training set

```
set.seed(9, sample.kind = "Rounding")
train_glm <- train(train_x, as.factor(train_y),
                   method = "glm",
                   family = "binomial",
                   trControl = control)
```

creating the predictions

```
glm_preds <- predict(train_glm, test_x)
```

creating a confusion matrix

```
cm_glm <- confusionMatrix(glm_preds,test_y, positive = "M")
cm_glm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 68  2
##          M  4 41
##
##                Accuracy : 0.948
##                  95% CI : (0.89, 0.981)
##     No Information Rate : 0.626
##     P-Value [Acc > NIR] : 5.75e-16
##
##                   Kappa : 0.89
##
##  Mcnemar's Test P-Value : 0.683
##
##             Sensitivity : 0.953
##             Specificity : 0.944
##          Pos Pred Value : 0.911
##          Neg Pred Value : 0.971
```

```
##               Prevalence : 0.374
##           Detection Rate : 0.357
##     Detection Prevalence : 0.391
##        Balanced Accuracy : 0.949
##
##         'Positive' Class : M
##
```

4.3 random forest

training the model using the training set

```
set.seed(9, sample.kind = "Rounding")
train_rf <- train(train_x, train_y,
                  method = "rf",
                  tuneGrid = data.frame(mtry = seq(2,40,2)),
                  importance = TRUE,
                  trControl = control)
```

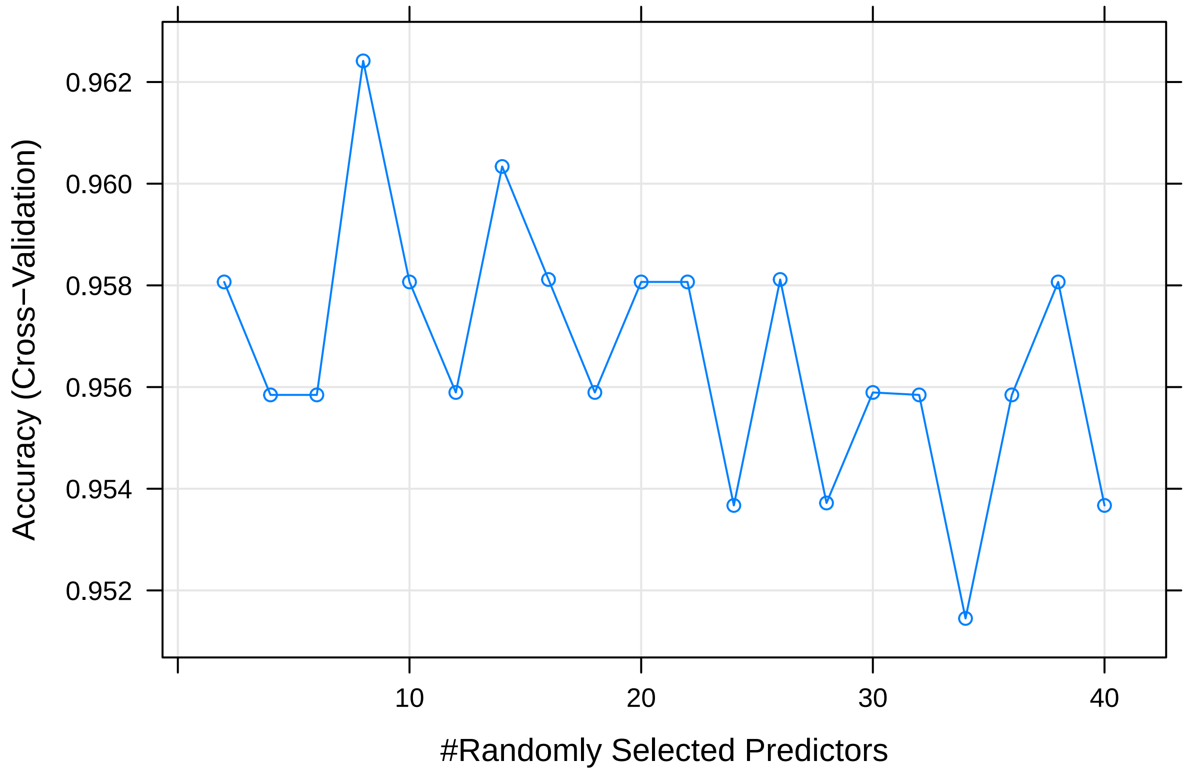
best tune

```
train_rf$bestTune
```

```
##   mtry
## 4    8
```

plot of training results

```
plot(train_rf)
```



predictions

```
rf_preds <- predict(train_rf, test_x)
```

variable importance

```
varImp(train_rf)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 30)
##
##                      Importance
## perimeter_worst           100.0
## radius_worst               98.1
## concave_pts_worst          95.5
## area_worst                 83.7
## concave_pts_mean           78.7
## area_se                    72.7
## texture_mean               69.6
## texture_worst              65.5
## concavity_worst            60.3
## concavity_mean             51.9
## radius_se                  51.6
## smoothness_worst           50.6
```

```
## perimeter_se            41.5
## radius_mean             30.4
## area_mean               29.4
## perimeter_mean          25.6
## compactness_worst       22.3
## symmetry_worst          20.8
## smoothness_mean         19.3
## concavity_se            18.9
```

creating a confusion matrix

```r
cm_rf <- confusionMatrix(rf_preds, test_y, positive = "M")
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 69  2
##          M  3 41
##
##                Accuracy : 0.957
##                  95% CI : (0.901, 0.986)
##     No Information Rate : 0.626
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.908
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.953
##             Specificity : 0.958
##          Pos Pred Value : 0.932
##          Neg Pred Value : 0.972
##              Prevalence : 0.374
##          Detection Rate : 0.357
##    Detection Prevalence : 0.383
##       Balanced Accuracy : 0.956
##
##        'Positive' Class : M
##
```

4.4 K Nearest Neighbors

setting up the tuning parameters

```r
set.seed(7, sample.kind = "Rounding")
tuning <- data.frame(k = seq(1, 20, 1))
```

training the model

```r
train_knn <- train(train_x, train_y,
                   method = "knn",
                   tuneGrid = tuning,
                   trControl = control)
```

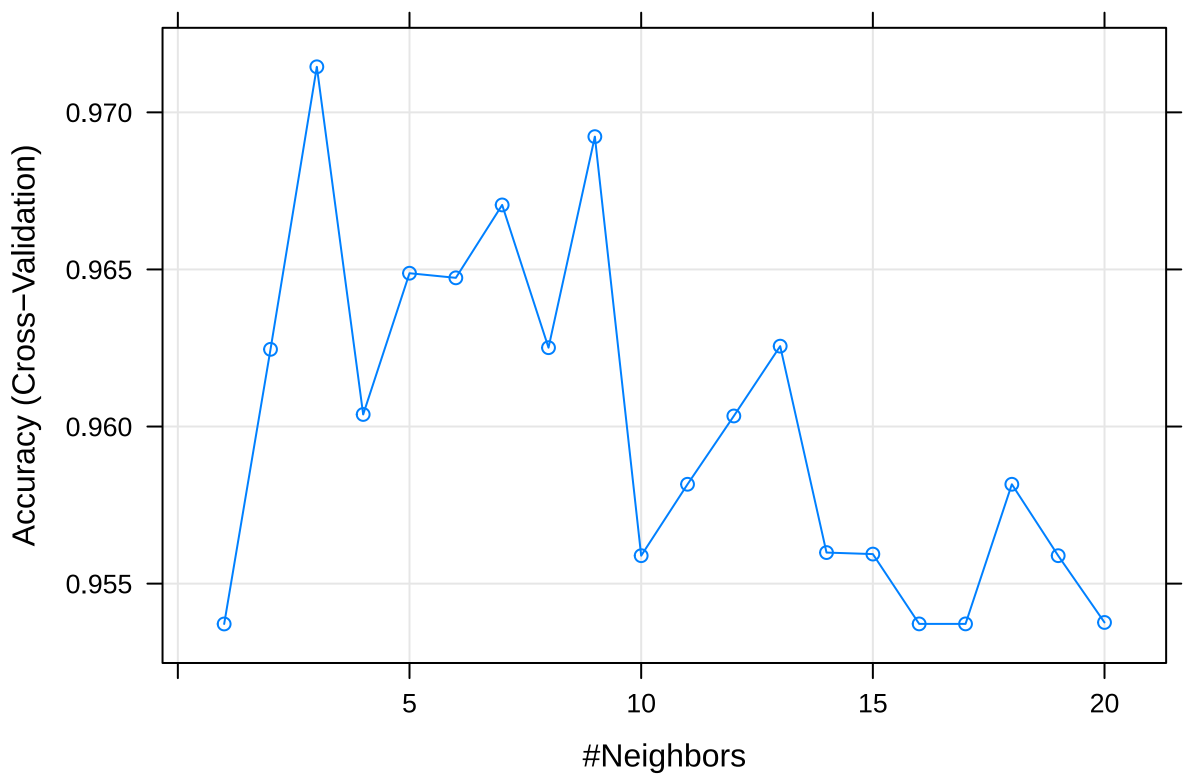
best tune

```r
train_knn$bestTune
```

```
##   k
## 3 3
```

plot of training model results

```r
plot(train_knn)
```



predictions

```r
knn_preds <- predict(train_knn, test_x)
```

creating a confusion matrix

```r
cm_knn <- confusionMatrix(knn_preds, test_y, positive = "M")
cm_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 71  1
##          M  1 42
##
##                Accuracy : 0.983
##                  95% CI : (0.939, 0.998)
##     No Information Rate : 0.626
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.963
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.977
##             Specificity : 0.986
##          Pos Pred Value : 0.977
##          Neg Pred Value : 0.986
##              Prevalence : 0.374
##          Detection Rate : 0.365
##    Detection Prevalence : 0.374
##       Balanced Accuracy : 0.981
##
##        'Positive' Class : M
##
```

4.5 Linear discriminant analysis

training the model using the training set

```r
set.seed(7, sample.kind = "Rounding")
train_lda <- train(train_x, train_y,
                   method = "lda",
                   trControl = control)
```

predictions

```r
lda_preds <- predict(train_lda, test_x)
```

creating a confusion matrix

```r
cm_LDA <- confusionMatrix(lda_preds, test_y,
                          positive = "M")
cm_LDA
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 70  5
##          M  2 38
##
##                Accuracy : 0.939
##                  95% CI : (0.879, 0.975)
##     No Information Rate : 0.626
##     P-Value [Acc > NIR] : 5.45e-15
##
##                   Kappa : 0.868
##
##  Mcnemar's Test P-Value : 0.45
##
##             Sensitivity : 0.884
##             Specificity : 0.972
##          Pos Pred Value : 0.950
##          Neg Pred Value : 0.933
##              Prevalence : 0.374
##          Detection Rate : 0.330
##    Detection Prevalence : 0.348
##       Balanced Accuracy : 0.928
##
##        'Positive' Class : M
##
```

4.6 Neural Network

setting the tuning parameter size and decay

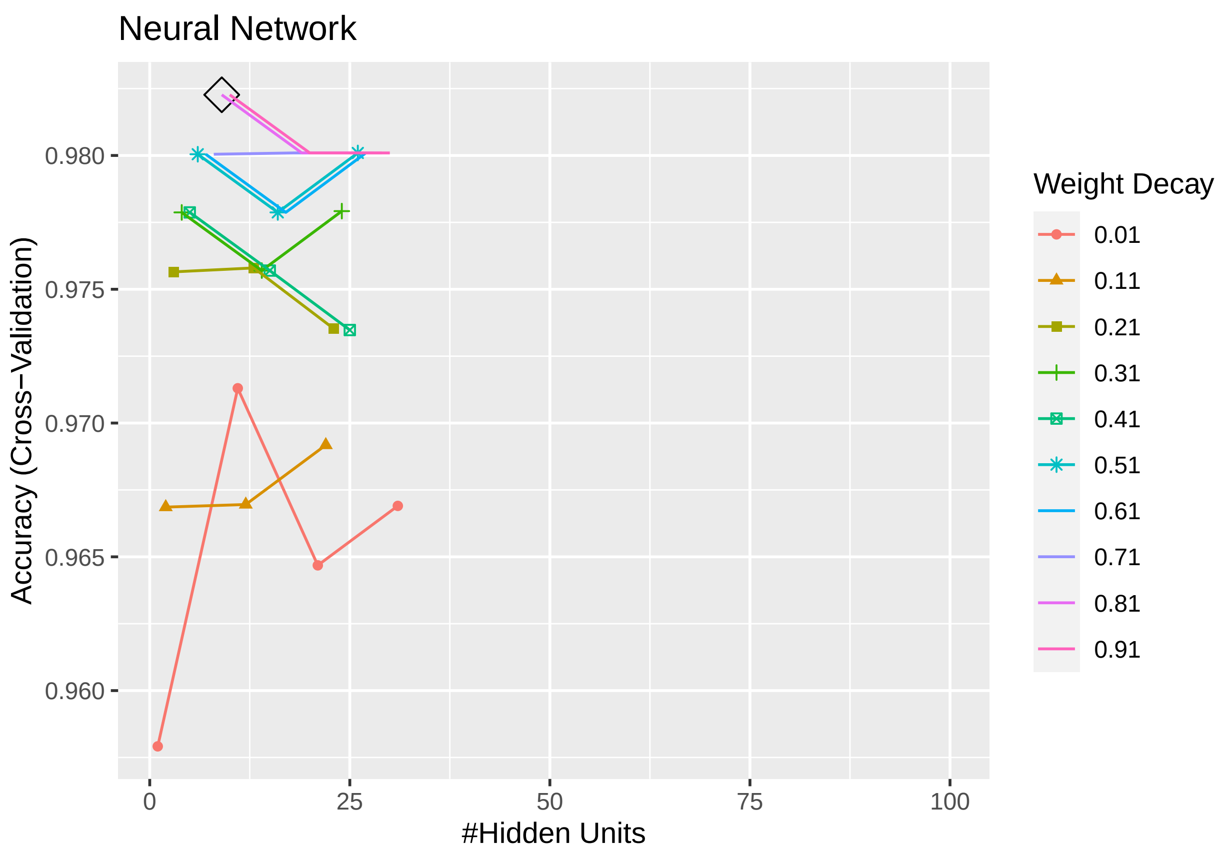
```r
set.seed(7, sample.kind = "Rounding")

tuning <- data.frame(size = seq(100), decay = seq(.01,1,.1))
```

training the model on the train set

```r
train_nn <- train(train_x, train_y,
                  method = "nnet",
                  tuneGrid = tuning,
                  trControl = control)
```

creating a graph for the tuning results

```r
ggplot(train_nn, highlight = TRUE) +
  ggtitle("Neural Network")
```



best tune

```
train_nn$bestTune
```

```
##   size decay
## 9    9  0.81
```

creating predictions

```
nn_preds <- predict(train_nn, test_x)
```

getting accuracy results

```
cm_nn <- confusionMatrix(nn_preds, test_y, positive = "M")
```

viewing accuracy results

```
cm_nn
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   B  M
##            B 71  2
```

```
##          M  1 41
##
##                Accuracy : 0.974
##                  95% CI : (0.926, 0.995)
##     No Information Rate : 0.626
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.944
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.953
##             Specificity : 0.986
##          Pos Pred Value : 0.976
##          Neg Pred Value : 0.973
##              Prevalence : 0.374
##          Detection Rate : 0.357
##    Detection Prevalence : 0.365
##       Balanced Accuracy : 0.970
##
##        'Positive' Class : M
##
```

4.7 Ensemble

creating a data frame of the prediction results of all the models

```
preds <- data.frame(log_r = glm_preds,
                    rf = rf_preds,
                    knn = knn_preds,
                    lda = lda_preds,
                    nn = nn_preds)
preds
```

```
##     log_r rf knn lda nn
## 1       B  B   B   B  B
## 2       B  B   B   B  B
## 3       B  B   B   B  B
## 4       B  B   B   B  B
## 5       B  B   B   B  B
## 6       B  M   M   M  B
## 7       B  M   B   B  B
## 8       B  B   B   B  B
## 9       B  B   B   B  B
## 10      M  B   B   B  B
## 11      B  B   B   B  B
## 12      B  B   B   B  B
## 13      B  B   B   B  B
## 14      B  B   B   B  B
## 15      B  B   B   B  B
## 16      B  B   B   B  B
## 17      B  B   B   B  B
```

```
## 18        B  B   B   B  B
## 19        B  B   B   B  B
## 20        B  B   B   B  B
## 21        B  B   B   B  B
## 22        B  B   B   B  B
## 23        B  B   B   B  B
## 24        B  B   B   B  B
## 25        B  B   B   B  B
## 26        B  B   B   B  B
## 27        B  B   B   B  B
## 28        B  B   B   B  B
## 29        B  B   B   B  B
## 30        B  B   B   B  B
## 31        M  B   B   B  B
## 32        B  B   B   B  B
## 33        B  B   B   B  B
## 34        B  B   B   B  B
## 35        B  B   B   B  B
## 36        B  B   B   B  B
## 37        B  B   B   B  B
## 38        B  B   B   B  B
## 39        B  B   B   B  B
## 40        B  B   B   B  B
## 41        B  B   B   B  B
## 42        B  B   B   B  B
## 43        B  B   B   B  B
## 44        B  B   B   B  B
## 45        B  B   B   B  B
## 46        B  B   B   B  B
## 47        B  B   B   B  B
## 48        B  B   B   B  B
## 49        B  B   B   B  B
## 50        B  M   B   B  B
## 51        B  B   B   B  B
## 52        B  B   B   B  B
## 53        B  B   B   B  B
## 54        B  B   B   B  B
## 55        B  B   B   B  B
## 56        B  B   B   B  B
## 57        B  B   B   B  B
## 58        B  B   B   B  B
## 59        B  B   B   B  B
## 60        M  B   B   B  B
## 61        B  B   B   B  B
## 62        B  B   B   B  B
## 63        B  B   B   B  B
## 64        B  B   B   B  B
## 65        B  B   B   B  B
## 66        M  B   B   M  M
## 67        B  B   B   B  B
## 68        B  B   B   B  B
## 69        B  B   B   B  B
## 70        B  B   B   B  B
## 71        B  B   B   B  B
```

```
## 72       B  B   B   B  B
## 73       M  M   M   M  M
## 74       M  M   M   M  M
## 75       M  M   M   M  M
## 76       M  M   M   M  M
## 77       M  M   M   M  M
## 78       M  M   M   M  M
## 79       M  M   M   M  M
## 80       M  M   M   M  M
## 81       M  M   M   M  M
## 82       M  M   M   M  M
## 83       B  B   B   B  B
## 84       M  M   M   M  M
## 85       M  M   M   M  M
## 86       M  M   M   M  M
## 87       M  M   M   M  M
## 88       M  M   M   M  M
## 89       M  M   M   M  M
## 90       M  M   M   M  B
## 91       M  M   M   M  M
## 92       M  B   M   M  M
## 93       B  M   M   B  M
## 94       M  M   M   M  M
## 95       M  M   M   M  M
## 96       M  M   M   B  M
## 97       M  M   M   M  M
## 98       M  M   M   M  M
## 99       M  M   M   M  M
## 100      M  M   M   B  M
## 101      M  M   M   M  M
## 102      M  M   M   M  M
## 103      M  M   M   M  M
## 104      M  M   M   M  M
## 105      M  M   M   M  M
## 106      M  M   M   M  M
## 107      M  M   M   B  M
## 108      M  M   M   M  M
## 109      M  M   M   M  M
## 110      M  M   M   M  M
## 111      M  M   M   M  M
## 112      M  M   M   M  M
## 113      M  M   M   M  M
## 114      M  M   M   M  M
## 115      M  M   M   M  M
```

Now that we have a data frame with all the predictions, we will take the mode of each sample and use that result as the ensemble's prediction for each sample.

```r
ensemble <- apply(preds,1,function(x) names(which.max(table(x))))

#factoring the results
ensemble <- as.factor(ensemble)
```

creating a confusion matrix

```r
cm_ensemble <- confusionMatrix(ensemble,
                               test_y,
                               positive = "M")
cm_ensemble
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 70  1
##          M  2 42
##
##                Accuracy : 0.974
##                  95% CI : (0.926, 0.995)
##     No Information Rate : 0.626
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.945
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.977
##             Specificity : 0.972
##          Pos Pred Value : 0.955
##          Neg Pred Value : 0.986
##              Prevalence : 0.374
##          Detection Rate : 0.365
##    Detection Prevalence : 0.383
##       Balanced Accuracy : 0.974
##
##        'Positive' Class : M
##
```

5. Results

5.1 Results table

```r
cm_list <- list(log_r = cm_glm,
                rf = cm_rf,
                knn = cm_knn,
                lda = cm_LDA,
                nn = cm_nn,
                ensemble = cm_ensemble)

cm_results <- sapply(cm_list, function(x) x$byClass)

results_table <- kable(cm_results)
results_table
```

	log_r	rf	knn	lda	nn	ensemble
Sensitivity	0.953	0.953	0.977	0.884	0.953	0.977
Specificity	0.944	0.958	0.986	0.972	0.986	0.972
Pos Pred Value	0.911	0.932	0.977	0.950	0.976	0.955
Neg Pred Value	0.971	0.972	0.986	0.933	0.973	0.986
Precision	0.911	0.932	0.977	0.950	0.976	0.955
Recall	0.953	0.953	0.977	0.884	0.953	0.977
F1	0.932	0.943	0.977	0.916	0.965	0.966
Prevalence	0.374	0.374	0.374	0.374	0.374	0.374
Detection Rate	0.357	0.357	0.365	0.330	0.357	0.365
Detection Prevalence	0.391	0.383	0.374	0.348	0.365	0.383
Balanced Accuracy	0.949	0.956	0.981	0.928	0.970	0.974

5.2 Best model

Which model had the highest Sensitivity?

```
which.max(cm_results[1,])
```

```
## knn
##   3
```

Which model had the highest Specificity?

```
which.max(cm_results[2,])
```

```
## knn
##   3
```

Which model had the highest F1 Score?

```
which.max(cm_results[7,])
```

```
## knn
##   3
```

Which model had the highest Balanced Accuracy?

```
which.max(cm_results[11,])
```

```
## knn
##   3
```

Knn is our best model by multiple performance measures

```
cm_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 71  1
##          M  1 42
##
##                Accuracy : 0.983
##                  95% CI : (0.939, 0.998)
##     No Information Rate : 0.626
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.963
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.977
##             Specificity : 0.986
##          Pos Pred Value : 0.977
##          Neg Pred Value : 0.986
##              Prevalence : 0.374
##          Detection Rate : 0.365
##    Detection Prevalence : 0.374
##       Balanced Accuracy : 0.981
##
##        'Positive' Class : M
##
```

6. Conclusion

6.1 summary

We were able to create six different models that were able to classify the data set into malignant and benign, including an ensemble which combined the results of the first five models. Out of the six models, the most accurate was the K Nearest Neighbors model with a F1 score of ***.977***

6.2 limitations

The main limitation of this project is that the size of this data set is small. Would these models hold up to such a high accuracy on a big data set?

6.3 future work

In my opinion this is a great starting place for predicting whether or not tumor samples are cancerous. In order to build on this model we would need to add tens of thousands of more samples and possibly more features. There might be other factors that doctors and researchers have found to be important such as family medical history, age, drugs or alcohol use, etc. Those might be relevant features to add to the data set. But all in all, this model is a great starting point for continuous breast cancer research.