Wisconsin Breast Cancer (Diagnostic) DataSet Analysis

Kalshtein Yael

22 בדצמבר 2017

Data Introduction

Identify the problem

Breast cancer is the most common malignancy among women, accounting for nearly 1 in 3 cancers diagnosed among women in the United States, and it is the second leading cause of cancer death among women. Breast Cancer occurs as a results of abnormal growth of cells in the breast tissue, commonly referred to as a Tumor. A tumor does not mean cancer - tumors can be benign (not cancerous), pre-malignant (pre-cancerous), or malignant (cancerous). Tests such as MRI, mammogram, ultrasound and biopsy are commonly used to diagnose breast cancer performed.

Identify data sources

This is an analysis of the Breast Cancer Wisconsin (Diagnostic) DataSet, obtained from Kaggle. This data set was created by Dr. William H. Wolberg, physician at the University Of Wisconsin Hospital at Madison, Wisconsin,USA. To create the dataset Dr. Wolberg used fluid samples, taken from patients with solid breast masses and an easy-to-use graphical computer program called Xcyt, which is capable of perform the analysis of cytological features based on a digital scan. The program uses a curve-fitting algorithm, to compute ten features from each one of the cells in the sample, than it calculates the mean value, extreme value and standard error of each feature for the image, returning a 30 real-valuated vector

Attribute Information:

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

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

a. radius (mean of distances from center to points on the perimeter)
b. texture (standard deviation of gray-scale values)
c. perimeter
d. area
e. smoothness (local variation in radius lengths)
f. compactness (perimeter^2 / area - 1.0)
g. concavity (severity of concave portions of the contour)
h. concave points (number of concave portions of the contour)
i. symmetry
j. fractal dimension (“coastline approximation” - 1)

The mean, standard error and “worst” or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

Objectives

This analysis aims to observe which features are most helpful in predicting malignant or benign cancer and to see general trends that may aid us in model selection and hyper parameter selection. The goal is to classify whether the breast cancer is benign or malignant. To achieve this i have used machine learning classification methods to fit a function that can predict the discrete class of new input.

```
#load libraries
library("ggplot2")
library("e1071")
library(dplyr)
library(reshape2)
library(corrplot)
library(caret)
library(pROC)
library(gridExtra)
library(grid)
library(ggfortify)
library(purrr)
library(nnet)
library(doParallel) # parallel processing
registerDoParallel()
require(foreach)
require(iterators)
require(parallel)
```

```
#Loading raw Data set
Cancer.rawdata <- read.csv("C:/Users/Yael/Desktop/R project/Breast Cancer Wisconsin.c
sv", sep=",")
```

Descriptive statistics

The first step is to visually inspect the new data set.

```
# Getting descriptive statistics
str(Cancer.rawdata)
```

```
'data.frame':   569 obs. of  33 variables:
 $ id                     : int  842302 842517 84300903 84348301 84358402 843786 8443
59 84458202 844981 84501001 ...
 $ diagnosis              : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
 $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
 $ texture_mean           : num  10.4 17.8 21.2 20.4 14.3 ...
 $ perimeter_mean         : num  122.8 132.9 130 77.6 135.1 ...
 $ area_mean              : num  1001 1326 1203 386 1297 ...
 $ smoothness_mean        : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
 $ compactness_mean       : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
 $ concavity_mean         : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
 $ concave.points_mean    : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
 $ symmetry_mean          : num  0.242 0.181 0.207 0.26 0.181 ...
 $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
 $ radius_se              : num  1.095 0.543 0.746 0.496 0.757 ...
 $ texture_se             : num  0.905 0.734 0.787 1.156 0.781 ...
 $ perimeter_se           : num  8.59 3.4 4.58 3.44 5.44 ...
 $ area_se                : num  153.4 74.1 94 27.2 94.4 ...
 $ smoothness_se          : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
 $ compactness_se         : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
 $ concavity_se           : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
 $ concave.points_se      : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
 $ symmetry_se            : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
 $ fractal_dimension_se   : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
 $ radius_worst           : num  25.4 25 23.6 14.9 22.5 ...
 $ texture_worst          : num  17.3 23.4 25.5 26.5 16.7 ...
 $ perimeter_worst        : num  184.6 158.8 152.5 98.9 152.2 ...
 $ area_worst             : num  2019 1956 1709 568 1575 ...
 $ smoothness_worst       : num  0.162 0.124 0.144 0.21 0.137 ...
 $ compactness_worst      : num  0.666 0.187 0.424 0.866 0.205 ...
 $ concavity_worst        : num  0.712 0.242 0.45 0.687 0.4 ...
 $ concave.points_worst   : num  0.265 0.186 0.243 0.258 0.163 ...
 $ symmetry_worst         : num  0.46 0.275 0.361 0.664 0.236 ...
 $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
 $ X                      : logi  NA NA NA NA NA NA ...
```

Id column is redundant and not useful, I would like to drop it.

Unnamed: 33 feature includes NaN so I will drop this one too.

```
#Remove the first  column
bc_data <- Cancer.rawdata[,-c(0:1)]
#Remove the last column
bc_data <- bc_data[,-32]
#Tidy the data
bc_data$diagnosis <- as.factor(bc_data$diagnosis)

head(bc_data)
```

```
  diagnosis radius_mean texture_mean perimeter_mean area_mean
1         M       17.99        10.38         122.80    1001.0
2         M       20.57        17.77         132.90    1326.0
3         M       19.69        21.25         130.00    1203.0
4         M       11.42        20.38          77.58     386.1
5         M       20.29        14.34         135.10    1297.0
6         M       12.45        15.70          82.57     477.1
  smoothness_mean compactness_mean concavity_mean concave.points_mean
1         0.11840          0.27760         0.3001             0.14710
2         0.08474          0.07864         0.0869             0.07017
3         0.10960          0.15990         0.1974             0.12790
4         0.14250          0.28390         0.2414             0.10520
5         0.10030          0.13280         0.1980             0.10430
6         0.12780          0.17000         0.1578             0.08089
  symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
1        0.2419                0.07871    1.0950     0.9053        8.589
2        0.1812                0.05667    0.5435     0.7339        3.398
3        0.2069                0.05999    0.7456     0.7869        4.585
4        0.2597                0.09744    0.4956     1.1560        3.445
5        0.1809                0.05883    0.7572     0.7813        5.438
6        0.2087                0.07613    0.3345     0.8902        2.217
  area_se smoothness_se compactness_se concavity_se concave.points_se
1  153.40      0.006399        0.04904      0.05373           0.01587
2   74.08      0.005225        0.01308      0.01860           0.01340
3   94.03      0.006150        0.04006      0.03832           0.02058
4   27.23      0.009110        0.07458      0.05661           0.01867
5   94.44      0.011490        0.02461      0.05688           0.01885
6   27.19      0.007510        0.03345      0.03672           0.01137
  symmetry_se fractal_dimension_se radius_worst texture_worst
1     0.03003             0.006193        25.38         17.33
2     0.01389             0.003532        24.99         23.41
3     0.02250             0.004571        23.57         25.53
4     0.05963             0.009208        14.91         26.50
5     0.01756             0.005115        22.54         16.67
6     0.02165             0.005082        15.47         23.75
  perimeter_worst area_worst smoothness_worst compactness_worst
1          184.60     2019.0           0.1622            0.6656
2          158.80     1956.0           0.1238            0.1866
3          152.50     1709.0           0.1444            0.4245
4           98.87      567.7           0.2098            0.8663
5          152.20     1575.0           0.1374            0.2050
6          103.40      741.6           0.1791            0.5249
  concavity_worst concave.points_worst symmetry_worst
1          0.7119               0.2654         0.4601
2          0.2416               0.1860         0.2750
3          0.4504               0.2430         0.3613
4          0.6869               0.2575         0.6638
5          0.4000               0.1625         0.2364
6          0.5355               0.1741         0.3985
  fractal_dimension_worst
1                 0.11890
2                 0.08902
3                 0.08758
4                 0.17300
5                 0.07678
6                 0.12440
```

Let’s check for missing variables:

```
#check for missing variables
sapply(bc_data, function(x) sum(is.na(x)))
```

```
             diagnosis            radius_mean           texture_mean 
                     0                      0                      0 
       perimeter_mean              area_mean        smoothness_mean 
                     0                      0                      0 
      compactness_mean        concavity_mean     concave.points_mean 
                     0                      0                      0 
        symmetry_mean  fractal_dimension_mean              radius_se 
                     0                      0                      0 
           texture_se            perimeter_se                 area_se 
                     0                      0                      0 
        smoothness_se         compactness_se           concavity_se 
                     0                      0                      0 
     concave.points_se             symmetry_se    fractal_dimension_se 
                     0                      0                      0 
         radius_worst           texture_worst         perimeter_worst 
                     0                      0                      0 
           area_worst        smoothness_worst       compactness_worst 
                     0                      0                      0 
       concavity_worst    concave.points_worst          symmetry_worst 
                     0                      0                      0 
fractal_dimension_worst 
                     0 
```

Missing values: none

Now that we have a good intuitive sense of the data, the next step involves taking a closer look at attributes and data values

```
summary(bc_data)
```

```
diagnosis  radius_mean      texture_mean   perimeter_mean
B:357     Min.   : 6.981   Min.   : 9.71   Min.   : 43.79
M:212     1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17
          Median :13.370   Median :18.84   Median : 86.24
          Mean   :14.127   Mean   :19.29   Mean   : 91.97
          3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10
          Max.   :28.110   Max.   :39.28   Max.   :188.50
  area_mean      smoothness_mean   compactness_mean  concavity_mean
Min.   : 143.5   Min.   :0.05263   Min.   :0.01938   Min.   :0.00000
1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492   1st Qu.:0.02956
Median : 551.1   Median :0.09587   Median :0.09263   Median :0.06154
Mean   : 654.9   Mean   :0.09636   Mean   :0.10434   Mean   :0.08880
3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040   3rd Qu.:0.13070
Max.   :2501.0   Max.   :0.16340   Max.   :0.34540   Max.   :0.42680
concave.points_mean symmetry_mean    fractal_dimension_mean
Min.   :0.00000     Min.   :0.1060   Min.   :0.04996
1st Qu.:0.02031     1st Qu.:0.1619   1st Qu.:0.05770
Median :0.03350     Median :0.1792   Median :0.06154
Mean   :0.04892     Mean   :0.1812   Mean   :0.06280
3rd Qu.:0.07400     3rd Qu.:0.1957   3rd Qu.:0.06612
Max.   :0.20120     Max.   :0.3040   Max.   :0.09744
  radius_se        texture_se      perimeter_se       area_se
Min.   :0.1115   Min.   :0.3602   Min.   : 0.757   Min.   :  6.802
1st Qu.:0.2324   1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850
Median :0.3242   Median :1.1080   Median : 2.287   Median : 24.530
Mean   :0.4052   Mean   :1.2169   Mean   : 2.866   Mean   : 40.337
3rd Qu.:0.4789   3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190
Max.   :2.8730   Max.   :4.8850   Max.   :21.980   Max.   :542.200
smoothness_se      compactness_se      concavity_se
Min.   :0.001713   Min.   :0.002252   Min.   :0.00000
1st Qu.:0.005169   1st Qu.:0.013080   1st Qu.:0.01509
Median :0.006380   Median :0.020450   Median :0.02589
Mean   :0.007041   Mean   :0.025478   Mean   :0.03189
3rd Qu.:0.008146   3rd Qu.:0.032450   3rd Qu.:0.04205
Max.   :0.031130   Max.   :0.135400   Max.   :0.39600
concave.points_se   symmetry_se       fractal_dimension_se
Min.   :0.000000   Min.   :0.007882   Min.   :0.0008948
1st Qu.:0.007638   1st Qu.:0.015160   1st Qu.:0.0022480
Median :0.010930   Median :0.018730   Median :0.0031870
Mean   :0.011796   Mean   :0.020542   Mean   :0.0037949
3rd Qu.:0.014710   3rd Qu.:0.023480   3rd Qu.:0.0045580
Max.   :0.052790   Max.   :0.078950   Max.   :0.0298400
 radius_worst   texture_worst   perimeter_worst    area_worst
Min.   : 7.93   Min.   :12.02   Min.   : 50.41   Min.   : 185.2
1st Qu.:13.01   1st Qu.:21.08   1st Qu.: 84.11   1st Qu.: 515.3
Median :14.97   Median :25.41   Median : 97.66   Median : 686.5
Mean   :16.27   Mean   :25.68   Mean   :107.26   Mean   : 880.6
3rd Qu.:18.79   3rd Qu.:29.72   3rd Qu.:125.40   3rd Qu.:1084.0
Max.   :36.04   Max.   :49.54   Max.   :251.20   Max.   :4254.0
smoothness_worst  compactness_worst concavity_worst  concave.points_worst
Min.   :0.07117   Min.   :0.02729   Min.   :0.0000   Min.   :0.00000
1st Qu.:0.11660   1st Qu.:0.14720   1st Qu.:0.1145   1st Qu.:0.06493
Median :0.13130   Median :0.21190   Median :0.2267   Median :0.09993
Mean   :0.13237   Mean   :0.25427   Mean   :0.2722   Mean   :0.11461
3rd Qu.:0.14600   3rd Qu.:0.33910   3rd Qu.:0.3829   3rd Qu.:0.16140
Max.   :0.22260   Max.   :1.05800   Max.   :1.2520   Max.   :0.29100
symmetry_worst   fractal_dimension_worst
```

```
 Min.   :0.1565   Min.   :0.05504
 1st Qu.:0.2504   1st Qu.:0.07146
 Median :0.2822   Median :0.08004
 Mean   :0.2901   Mean   :0.08395
 3rd Qu.:0.3179   3rd Qu.:0.09208
 Max.   :0.6638   Max.   :0.20750
```

Description

In the results displayed, you can see the data has 569 records, each with 31 columns.

Diagnosis is a categorical variable.

All feature values are recoded with four significant digits.

Missing attribute values: none

Class distribution: 357 benign, 212 malignant

Univariate Plots Section

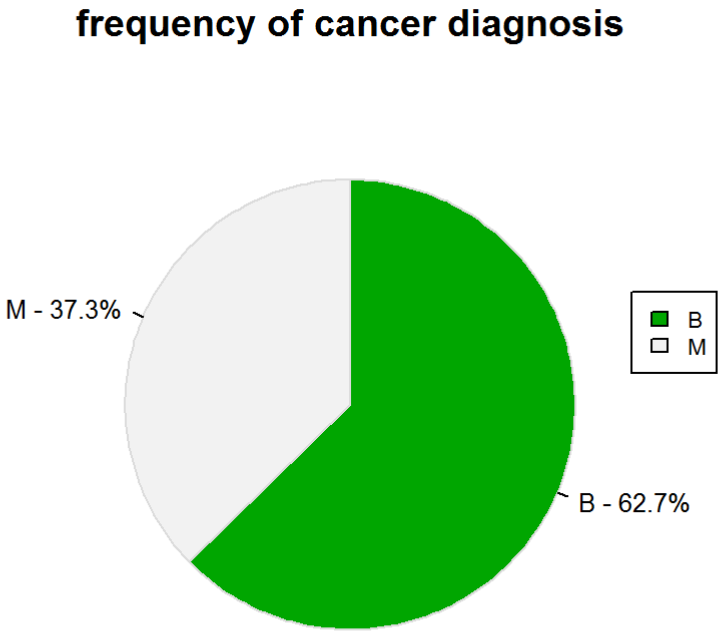
One of the main goals of visualizing the data here is to observe which features are most helpful in predicting malignant or benign cancer. The other is to see general trends that may aid us in model selection and hyper parameter selection.

I will analyze the features and try to understand which features have larger predictive value and which does not bring considerable predictive value if we want to create a model that allows us to guess if a tumor is benign or malignant.

frequency of cancer diagnosis

first lets get the frequency of cancer diagnosis

```
## Create a frequency table
diagnosis.table <- table(bc_data$diagnosis)
colors <- terrain.colors(2)
# Create a pie chart
diagnosis.prop.table <- prop.table(diagnosis.table)*100
diagnosis.prop.df <- as.data.frame(diagnosis.prop.table)
pielabels <- sprintf("%s - %3.1f%s", diagnosis.prop.df[,1], diagnosis.prop.table, "%"
)

pie(diagnosis.prop.table,
  labels=pielabels,
  clockwise=TRUE,
  col=colors,
  border="gainsboro",
  radius=0.8,
  cex=0.8,
  main="frequency of cancer diagnosis")
legend(1, .4, legend=diagnosis.prop.df[,1], cex = 0.7, fill = colors)
```



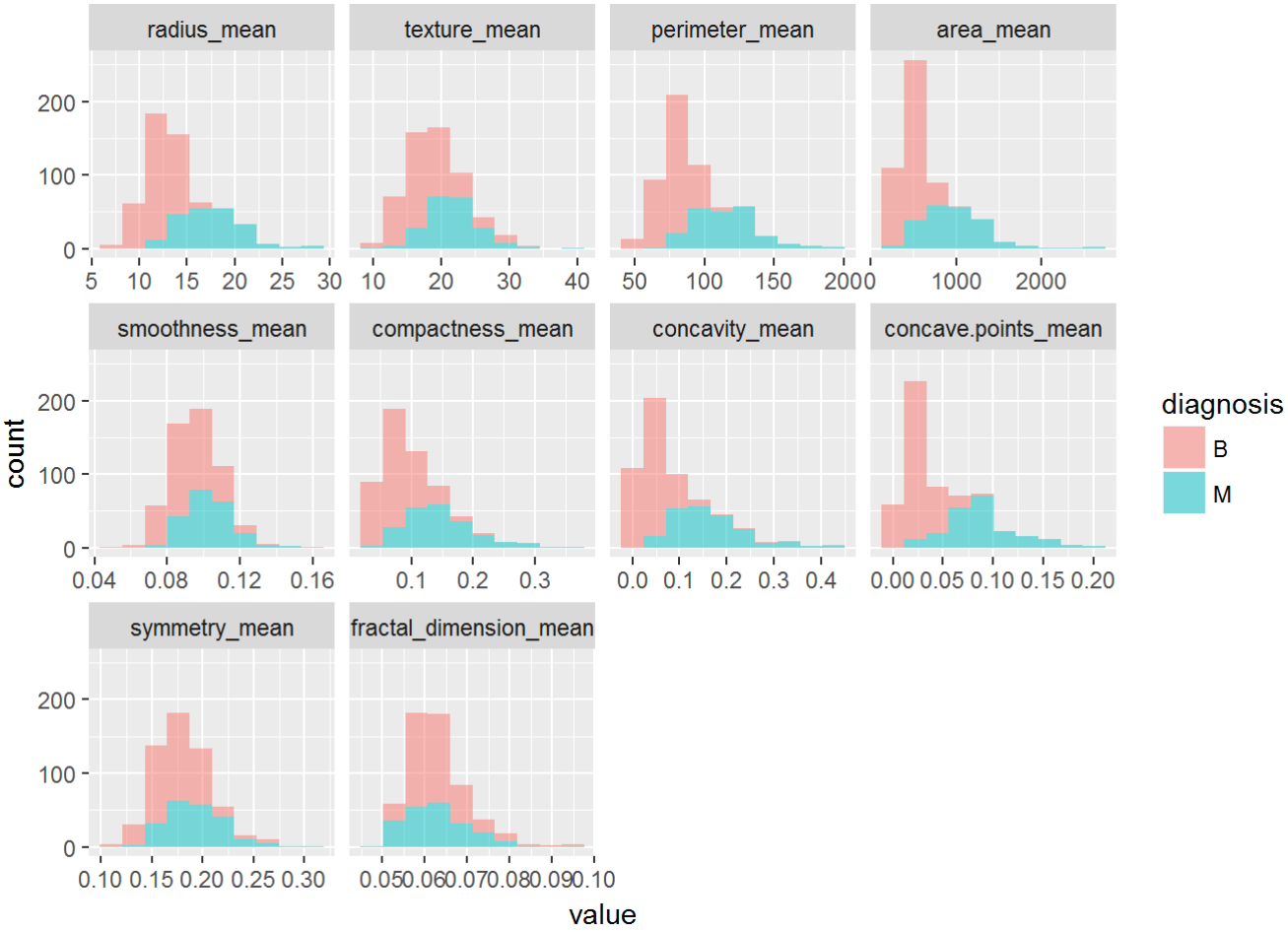
M= Malignant (indicates prescence of cancer cells); B= Benign (indicates abscence)

357 observations which account for 62.7% of all observations indicating the absence of cancer cells, 212 which account for 37.3% of all observations shows the presence of cancerous cell.

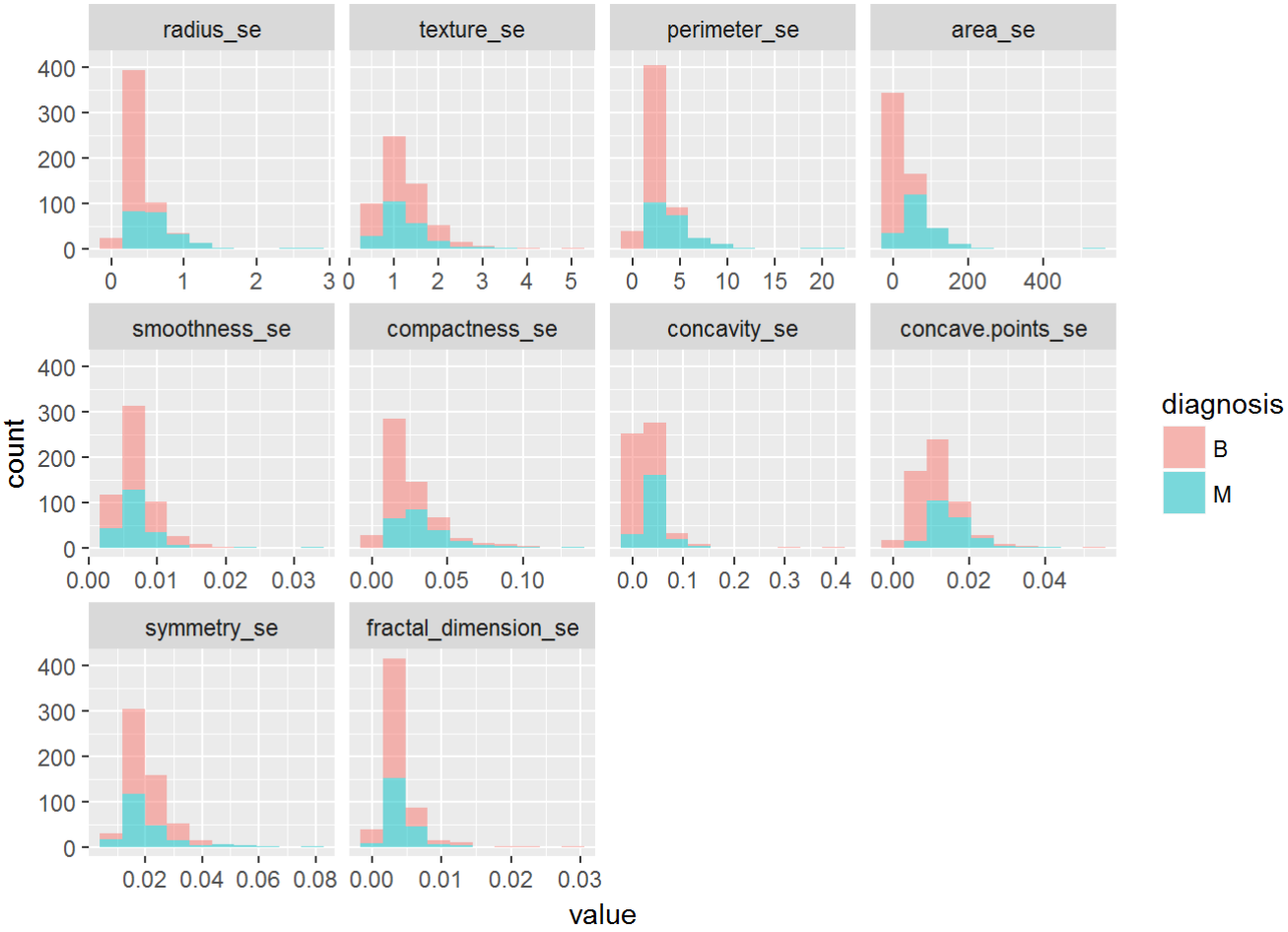
The percent is unusually large; the dataset does not represents in this case a typical medical analysis distribution. Typically, we will have a considerable large number of cases that represents negative vs. a small number of cases that represents positives (malignant) tumor.

Visualise distribution of data via histograms

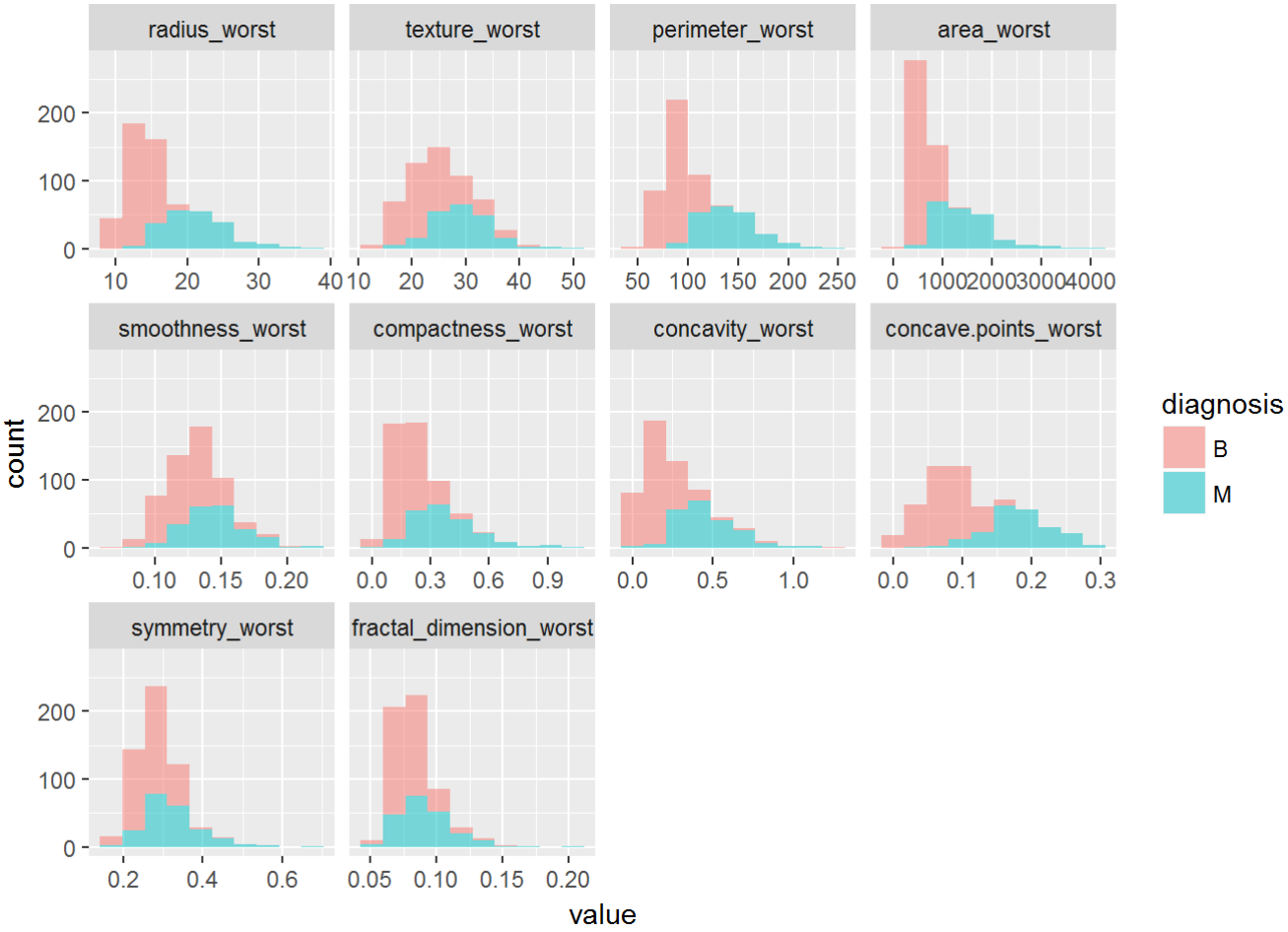
```
#Break up columns into groups, according to their suffix designation
#(_mean, _se,and __worst) to perform visualisation plots off.
data_mean <- Cancer.rawdata[ ,c("diagnosis", "radius_mean", "texture_mean","perimeter
_mean", "area_mean", "smoothness_mean", "compactness_mean", "concavity_mean", "concav
e.points_mean", "symmetry_mean", "fractal_dimension_mean" )]

data_se <- Cancer.rawdata[ ,c("diagnosis", "radius_se", "texture_se","perimeter_se",
"area_se", "smoothness_se", "compactness_se", "concavity_se", "concave.points_se", "s
ymmetry_se", "fractal_dimension_se" )]

data_worst <- Cancer.rawdata[ ,c("diagnosis", "radius_worst", "texture_worst","perime
ter_worst", "area_worst", "smoothness_worst", "compactness_worst", "concavity_worst",
"concave.points_worst", "symmetry_worst", "fractal_dimension_worst" )]

#Plot histograms of "_mean" variables group by diagnosis
ggplot(data = melt(data_mean, id.var = "diagnosis"), mapping = aes(x = value)) +
    geom_histogram(bins = 10, aes(fill=diagnosis), alpha=0.5) + facet_wrap(~variable,
scales =      'free_x')
```




```
#Plot histograms of "_se" variables group by diagnosis
ggplot(data = melt(data_se, id.var = "diagnosis"), mapping = aes(x = value)) +
    geom_histogram(bins = 10, aes(fill=diagnosis), alpha=0.5) + facet_wrap(~variable,
scales = 'free_x')
```




```
#Plot histograms of "_worst" variables group by diagnosis
ggplot(data = melt(data_worst, id.var = "diagnosis"), mapping = aes(x = value)) +
    geom_histogram(bins = 10, aes(fill=diagnosis), alpha=0.5) + facet_wrap(~variable,
scales = 'free_x')
```



Most of the features are normally distributed.

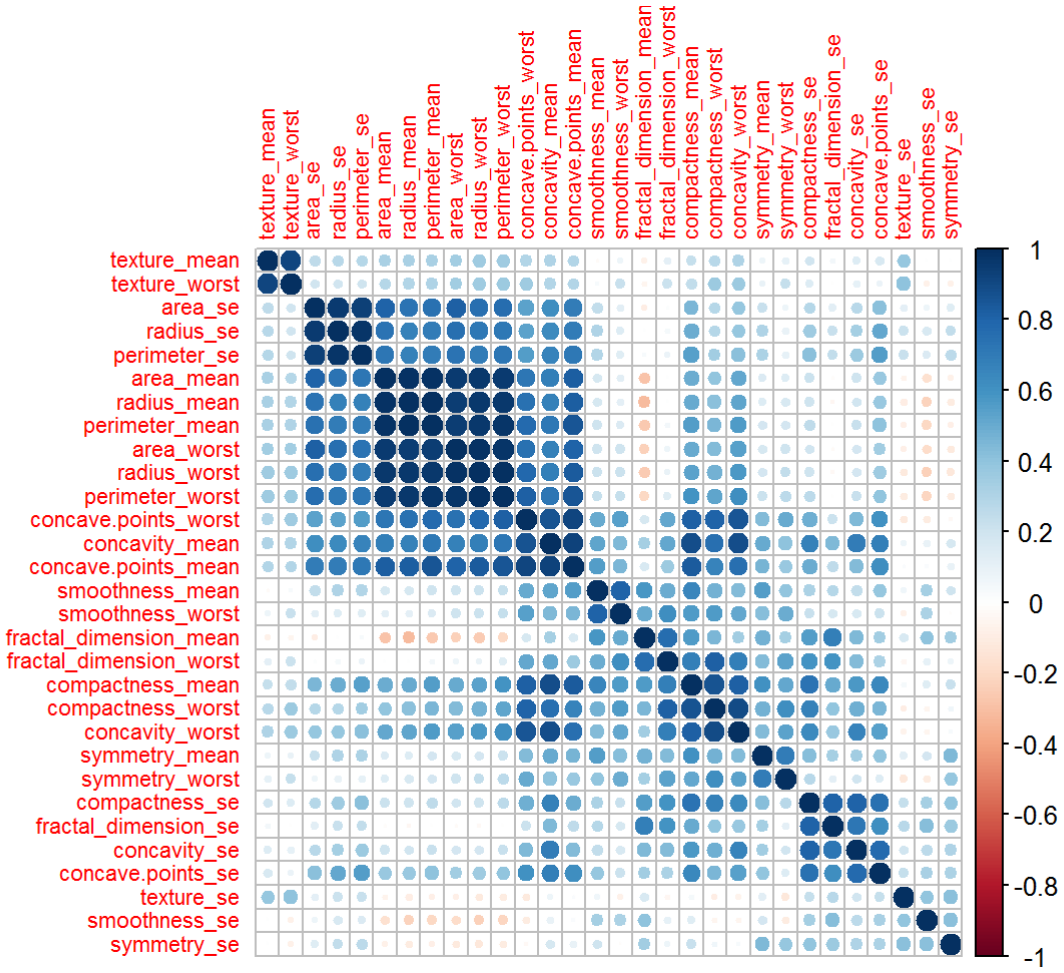
Comparison of radius distribution by malignancy shows that there is no perfect separation between any of the features; we do have fairly good separations for concave.points_worst, concavity_worst, perimeter_worst, area_mean, perimeter_mean. We do have as well tight superposition for some of the values, like symmetry_se, smoothness_se .

Bivariate/Multivariate Analysis

Correlation Plot

We are also interested in how the 30 predictors relate to each other. To see bivariate relationships among these 30 predictors, we will look at correlations.

```
# calculate collinearity
corMatMy <- cor(bc_data[,2:31])
corrplot(corMatMy, order = "hclust", tl.cex = 0.7)
```



There are quite a few variables that are correlated. Often we have features that are highly correlated and those provide redundant information. By eliminating highly correlated features we can avoid a predictive bias for the information contained in these features. This also shows us, that when we want to make statements about the biological/ medical importance of specific features, we need to keep in mind that just because they are suitable to predicting an outcome they are not necessarily causal - they could simply be correlated with causal factors.

I am now removing all features with a correlation higher than 0.9, keeping the feature with the lower mean.

```
highlyCor <- colnames(bc_data)[findCorrelation(corMatMy, cutoff = 0.9, verbose = TRUE
)]
```

```
Compare row 7  and column  8 with corr  0.921
  Means:  0.571 vs 0.389 so flagging column 7
Compare row 8  and column  28 with corr  0.91
  Means:  0.542 vs 0.377 so flagging column 8
Compare row 23  and column  21 with corr  0.994
  Means:  0.48 vs 0.367 so flagging column 23
Compare row 21  and column  3 with corr  0.969
  Means:  0.446 vs 0.359 so flagging column 21
Compare row 3  and column  24 with corr  0.942
  Means:  0.414 vs 0.353 so flagging column 3
Compare row 24  and column  1 with corr  0.941
  Means:  0.39 vs 0.349 so flagging column 24
Compare row 1  and column  4 with corr  0.987
  Means:  0.35 vs 0.347 so flagging column 1
Compare row 13  and column  11 with corr  0.973
  Means:  0.372 vs 0.346 so flagging column 13
Compare row 11  and column  14 with corr  0.952
  Means:  0.323 vs 0.347 so flagging column 14
Compare row 22  and column  2 with corr  0.912
  Means:  0.224 vs 0.357 so flagging column 2
All correlations <= 0.9
```

```
highlyCor
```

```
 [1] "compactness_mean"     "concavity_mean"       "texture_worst"
 [4] "fractal_dimension_se" "texture_mean"         "perimeter_worst"
 [7] "diagnosis"            "texture_se"           "perimeter_se"
[10] "radius_mean"
```

10 columns are flagged for removal.

```
bc_data_cor <- bc_data[, which(!colnames(bc_data) %in% highlyCor)]
ncol(bc_data_cor)
```

```
[1] 21
```

So our new data frame bc_data_cor is 10 variables shorter.

Data Preperation

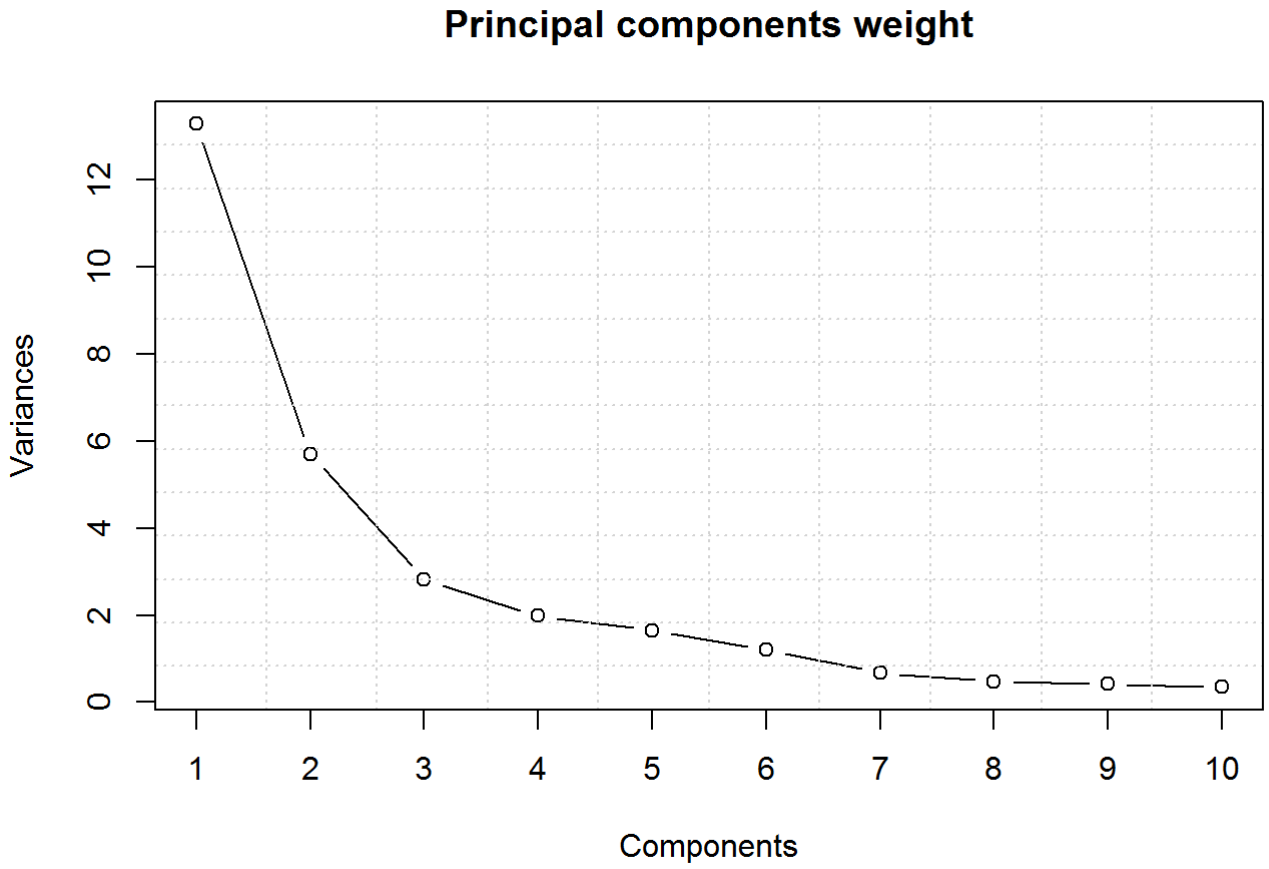
Data preperation is a crucial step for any data analysis problem. It is often a very good idea to prepare your data in such way to best expose the structure of the problem to the machine learning algorithms that you intend to use.

Because there are so much correlation some machine learning models can fail.In this section I am going to create a PCA version of the data

Principal Components Analysis (PCA) transform

PCA doesn't just center and rescale the individual variables. It constructs a set of orthogonal (non-collinear, uncorrelated, independent) variables. For many model fitting algorithms, these variables are much easier to fit than "natural" (somewhat collinear, somewhat correlated, not-independent) variables.

```
cancer.pca <- prcomp(bc_data[, 2:31], center=TRUE, scale=TRUE)
plot(cancer.pca, type="l", main='')
grid(nx = 10, ny = 14)
title(main = "Principal components weight", sub = NULL, xlab = "Components")
box()
```

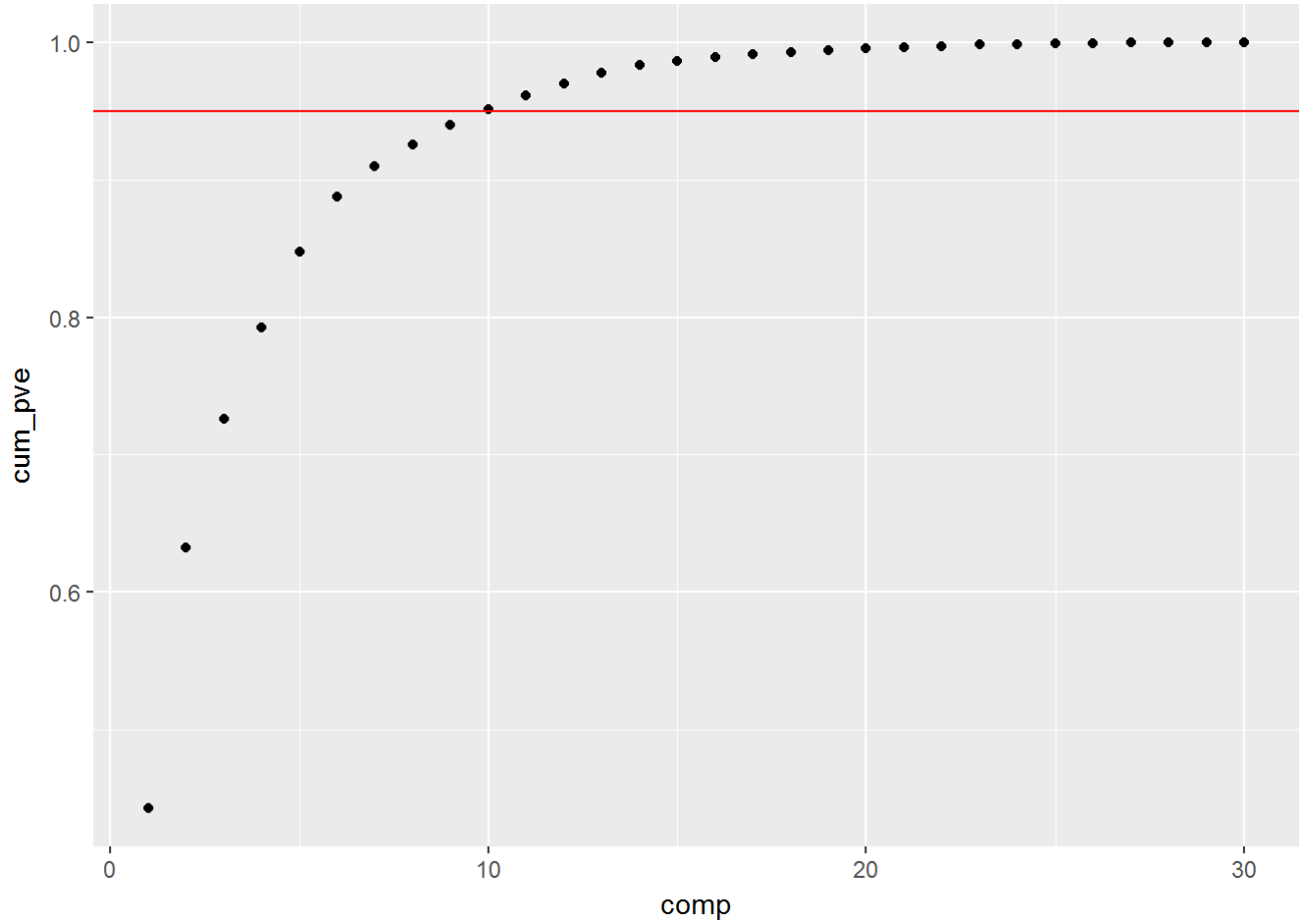



```
summary(cancer.pca)
```

```
Importance of components%s:
                          PC1    PC2     PC3     PC4     PC5     PC6
Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880
Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025
Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759
                           PC7     PC8    PC9    PC10   PC11    PC12
Standard deviation     0.82172 0.69037 0.6457 0.59219 0.5421 0.51104
Proportion of Variance 0.02251 0.01589 0.0139 0.01169 0.0098 0.00871
Cumulative Proportion  0.91010 0.92598 0.9399 0.95157 0.9614 0.97007
                          PC13    PC14    PC15    PC16    PC17    PC18
Standard deviation     0.49128 0.39624 0.30681 0.28260 0.24372 0.22939
Proportion of Variance 0.00805 0.00523 0.00314 0.00266 0.00198 0.00175
Cumulative Proportion  0.97812 0.98335 0.98649 0.98915 0.99113 0.99288
                          PC19    PC20   PC21    PC22    PC23   PC24
Standard deviation     0.22244 0.17652 0.1731 0.16565 0.15602 0.1344
Proportion of Variance 0.00165 0.00104 0.0010 0.00091 0.00081 0.0006
Cumulative Proportion  0.99453 0.99557 0.9966 0.99749 0.99830 0.9989
                          PC25    PC26    PC27    PC28    PC29    PC30
Standard deviation     0.12442 0.09043 0.08307 0.03987 0.02736 0.01153
Proportion of Variance 0.00052 0.00027 0.00023 0.00005 0.00002 0.00000
Cumulative Proportion  0.99942 0.99969 0.99992 0.99997 1.00000 1.00000
```

The two first components explains the 0.6324 of the variance.

```
# Calculate the proportion of variance explained
pca_var <- cancer.pca$sdev^2
pve_df <- pca_var / sum(pca_var)
cum_pve <- cumsum(pve_df)
pve_table <- tibble(comp = seq(1:ncol(bc_data %>% select(-diagnosis))), pve_df, cum_p
ve)

ggplot(pve_table, aes(x = comp, y = cum_pve)) +
  geom_point() +
  geom_abline(intercept = 0.95, color = "red", slope = 0)
```



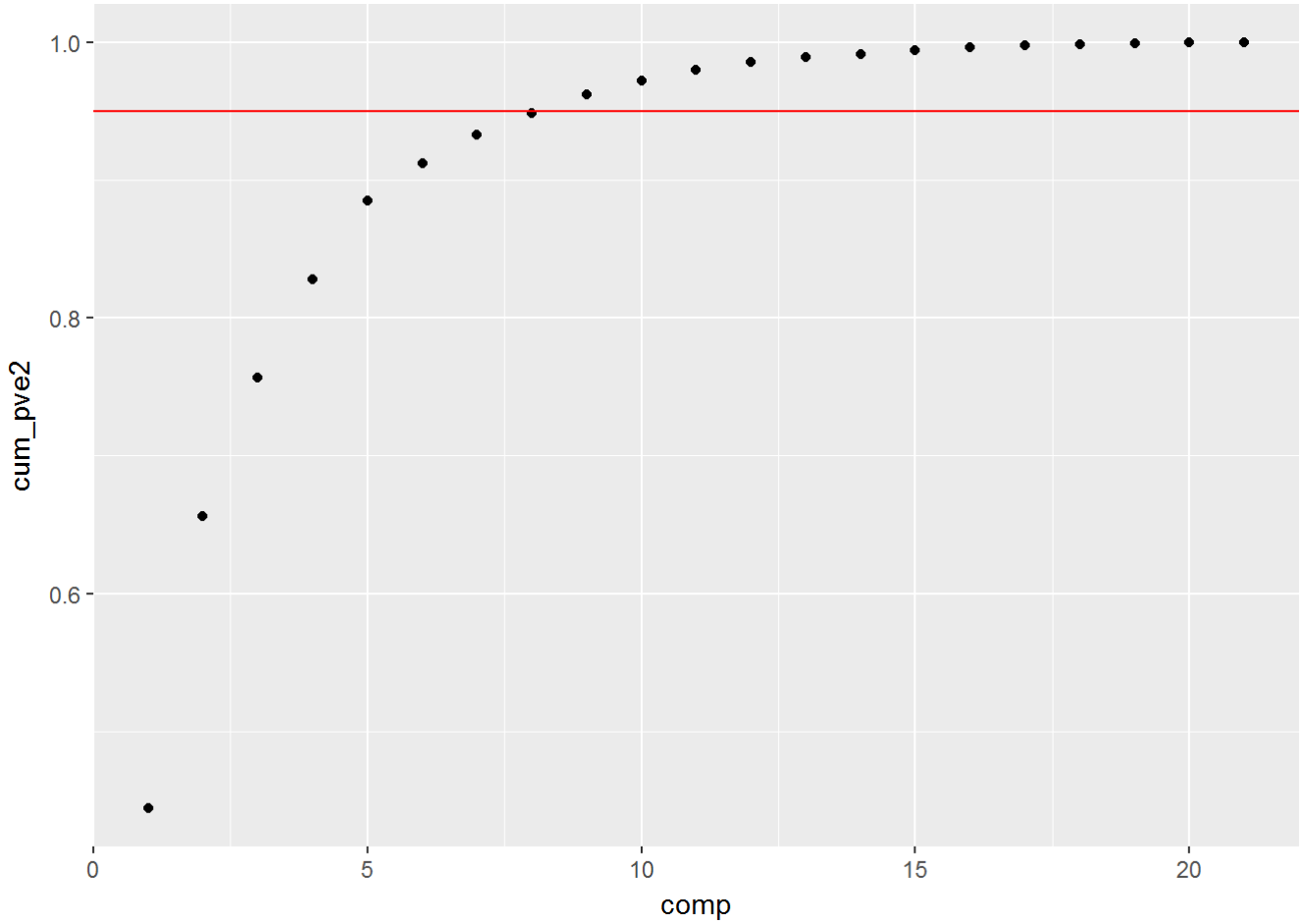
We need 10 principal components to explain more than 0.95 of the variance and 17 to explain more than 0.99.

Let's do the same exercise with our second df, the one where we removed the highly correlated predictors.

```
cancer.pca2 <- prcomp(bc_data_cor, center=TRUE, scale=TRUE)
summary(cancer.pca2)
```

```
Importance of components%s:
                          PC1    PC2   PC3     PC4     PC5     PC6     PC7
Standard deviation     3.053 2.1105 1.456 1.21994 1.09673 0.75004 0.66893
Proportion of Variance 0.444 0.2121 0.101 0.07087 0.05728 0.02679 0.02131
Cumulative Proportion  0.444 0.6561 0.757 0.82791 0.88519 0.91197 0.93328
                           PC8     PC9    PC10    PC11    PC12    PC13
Standard deviation     0.56454 0.53543 0.45639 0.41367 0.34423 0.26012
Proportion of Variance 0.01518 0.01365 0.00992 0.00815 0.00564 0.00322
Cumulative Proportion  0.94846 0.96211 0.97203 0.98018 0.98582 0.98904
                          PC14    PC15    PC16    PC17    PC18    PC19
Standard deviation     0.24137 0.22045 0.20547 0.17791 0.15094 0.13695
Proportion of Variance 0.00277 0.00231 0.00201 0.00151 0.00108 0.00089
Cumulative Proportion  0.99182 0.99413 0.99614 0.99765 0.99873 0.99963
                          PC20    PC21
Standard deviation     0.08384 0.02885
Proportion of Variance 0.00033 0.00004
Cumulative Proportion  0.99996 1.00000
```

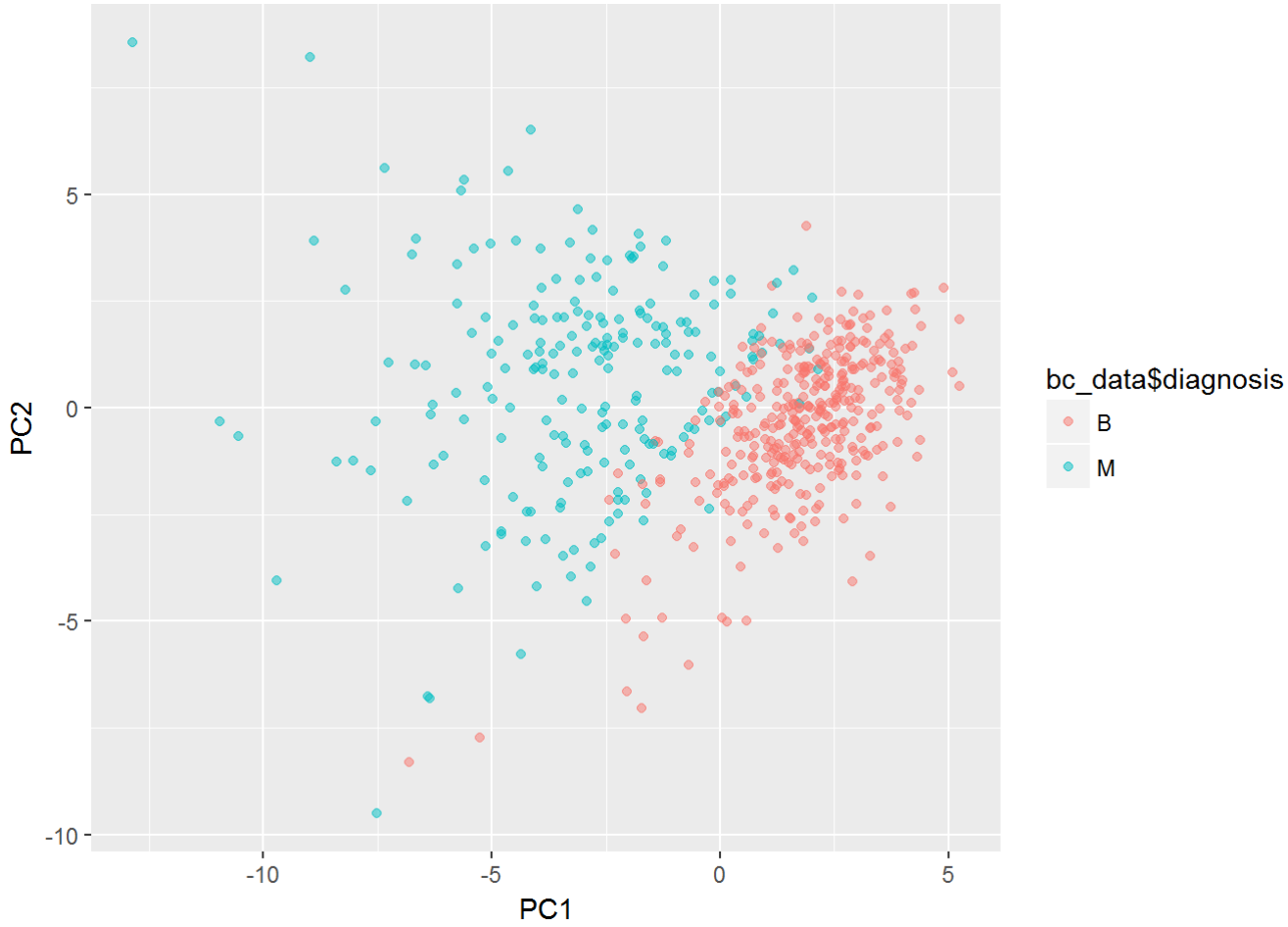
```
# Calculate the proportion of variance explained
pca_var2 <- cancer.pca2$sdev^2
pve_df2 <- pca_var2 / sum(pca_var2)
cum_pve2 <- cumsum(pve_df2)
pve_table2 <- tibble(comp = seq(1:ncol(bc_data_cor)), pve_df2, cum_pve2)

ggplot(pve_table2, aes(x = comp, y = cum_pve2)) +
  geom_point() +
  geom_abline(intercept = 0.95, color = "red", slope = 0)
```



In this case, around 8 PC's explained 95% of the variance and 13 PC'S explained more than 0.99%.

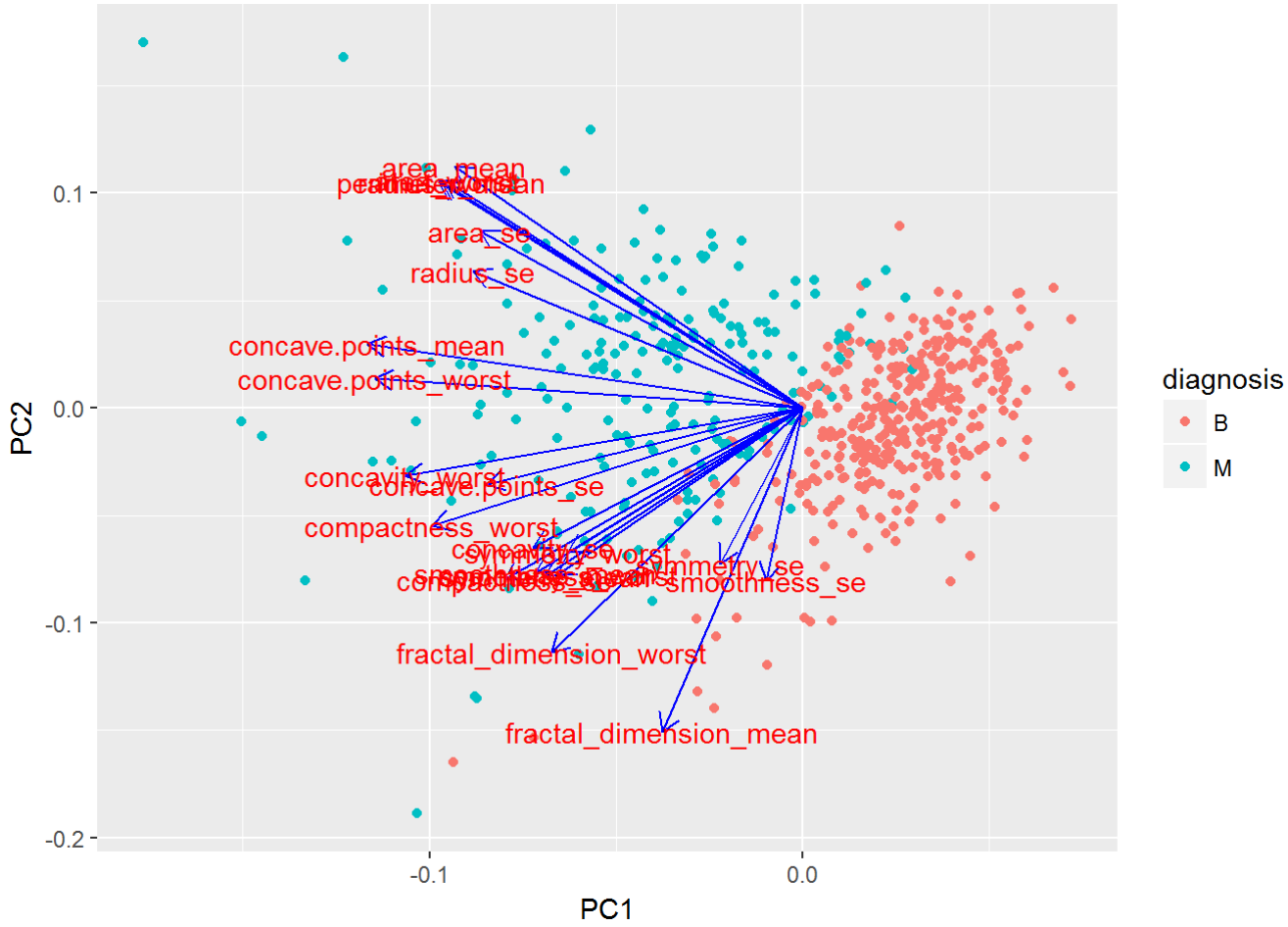
The features with highest dimmensions or aligned with the leading principal component are the ones with highest variance.

```
pca_df <- as.data.frame(cancer.pca2$x)
ggplot(pca_df, aes(x=PC1, y=PC2, col=bc_data$diagnosis)) + geom_point(alpha=0.5)
```



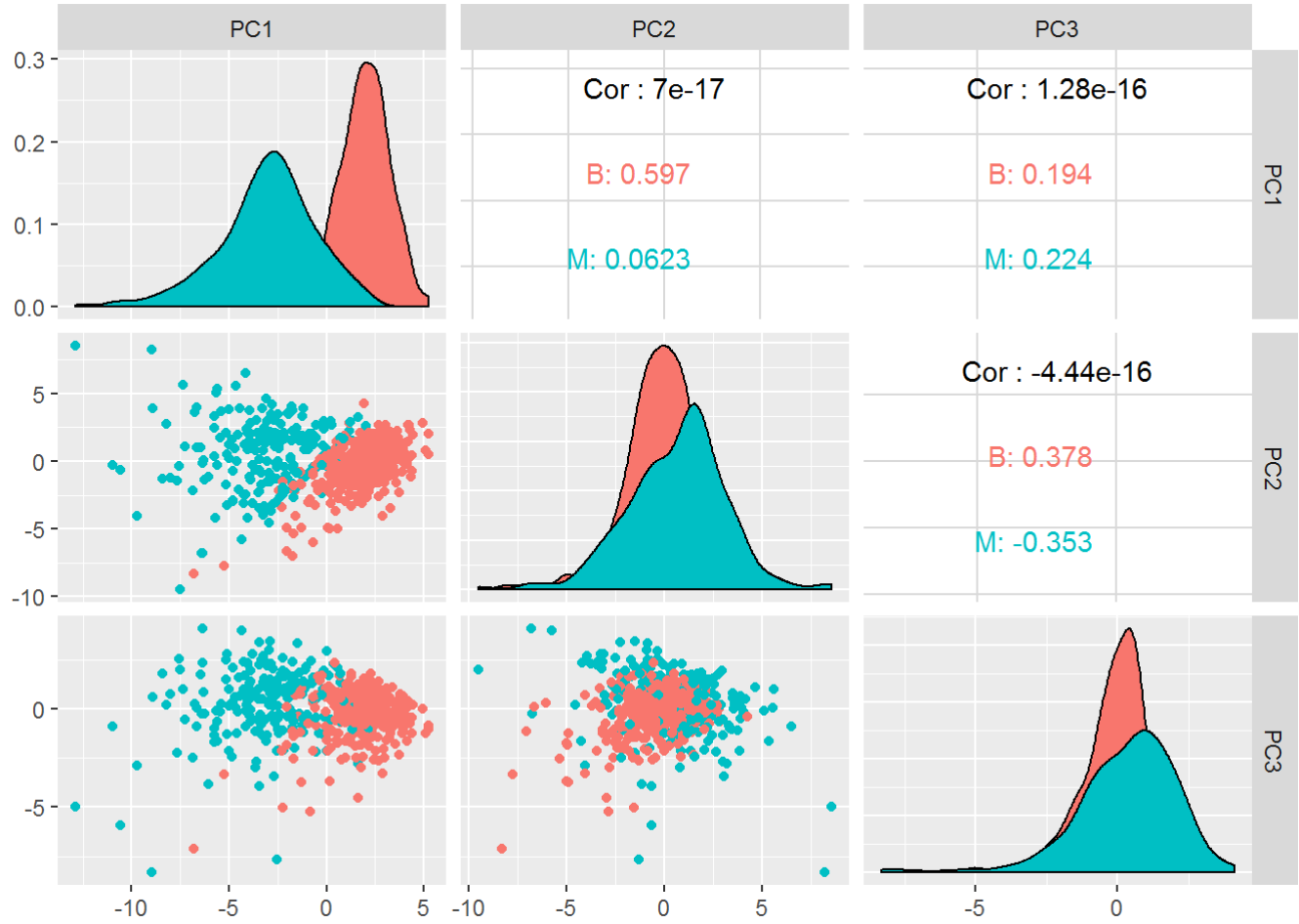
To visualize which variables are the most influential on the first 2 components

```
autoplot(cancer.pca2, data = bc_data,  colour = 'diagnosis',
                    loadings = FALSE, loadings.label = TRUE, loadings.colour = "blue"
)
```



Let's visuzalize the first 3 components.

```
df_pcs <- cbind(as_tibble(bc_data$diagnosis), as_tibble(cancer.pca2$x))
GGally::ggpairs(df_pcs, columns = 2:4, ggplot2::aes(color = value))
```



As it can be seen from the above plots the first 3 principal components separate the two classes to some extent only, this is expected since the variance explained by these components is not large.

***We will use the caret preProcess to apply pca with a 0.99 threshold

Split data into training and test sets

The simplest method to evaluate the performance of a machine learning algorithm is to use different training and testing datasets. I will Split the available data into a training set and a testing set. (70% training, 30% test)

```
#Split data set in train 70% and test 30%
set.seed(1234)
df <- cbind(diagnosis = bc_data$diagnosis, bc_data_cor)
train_indx <- createDataPartition(df$diagnosis, p = 0.7, list = FALSE)

train_set <- df[train_indx,]
test_set <- df[-train_indx,]

nrow(train_set)
```

```
[1] 399
```

```
nrow(test_set)
```

```
[1] 170
```

Applying machine learning models

In this section I will:

1.Train the algorithm on the first part,

2.make predictions on the second part and

3.evaluate the predictions against the expected results.

```
fitControl <- trainControl(method="cv",
                           number = 5,
                           preProcOptions = list(thresh = 0.99), # threshold for pca
preprocess
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)
```

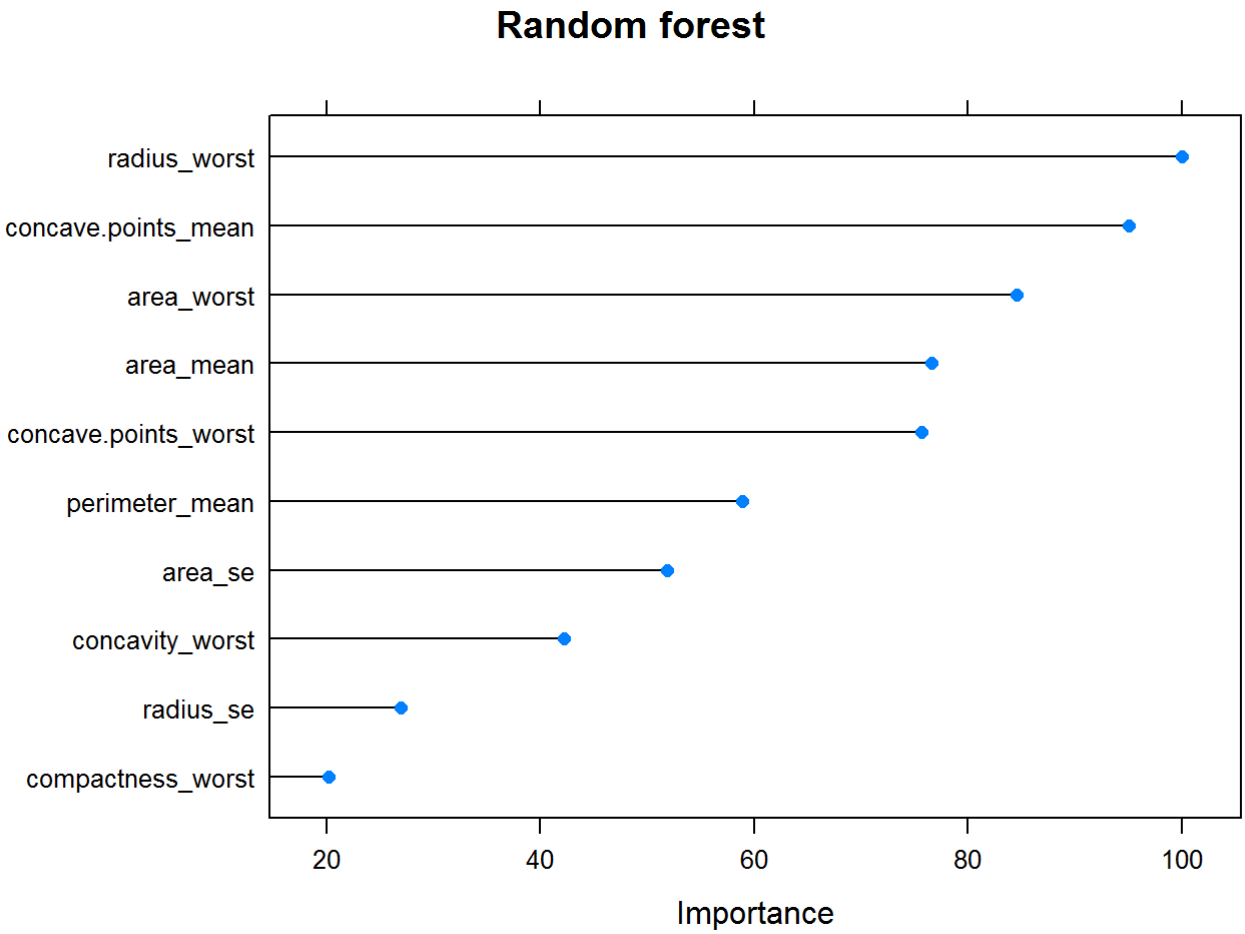
Random Forest

Let’s try random forest:

```
model_rf <- train(diagnosis~.,
                  data = train_set,
                  method="rf",
                  metric="ROC",
                  #tuneLength=10,
                  preProcess = c('center', 'scale'),
                  trControl=fitControl)
```

Let’s visualize the variable importance:

```
# plot feature importance
plot(varImp(model_rf), top = 10, main = "Random forest")
```



We observe that radius_worst, concave.points_mean, area_worst, area_mean, concave.points_worst, perimeter_mean, area_se and concavity_worst are the most important features. Most of them are also in the list of features with higher dimmension in the leading Principal Components plane or aligned with the leading Principal Component, PC1.

We present now the test data to the model.

```
pred_rf <- predict(model_rf, test_set)
cm_rf <- confusionMatrix(pred_rf, test_set$diagnosis, positive = "M")
cm_rf
```

```
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 103   6
         M   4  57

               Accuracy : 0.9412
                 95% CI : (0.8945, 0.9714)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8731
 Mcnemar's Test P-Value : 0.7518

            Sensitivity : 0.9048
            Specificity : 0.9626
         Pos Pred Value : 0.9344
         Neg Pred Value : 0.9450
             Prevalence : 0.3706
         Detection Rate : 0.3353
   Detection Prevalence : 0.3588
      Balanced Accuracy : 0.9337

       'Positive' Class : M
```

Random Forest with PCA

```
model_pca_rf <- train(diagnosis~.,
                  data = train_set,
                  method="ranger",
                  metric="ROC",
                  #tuneLength=10,
                  preProcess = c('center', 'scale', 'pca'),
                  trControl=fitControl)
```

```
pred_pca_rf <- predict(model_pca_rf, test_set)
cm_pca_rf <- confusionMatrix(pred_pca_rf, test_set$diagnosis, positive = "M")
cm_pca_rf
```

```
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 105   9
         M   2  54

               Accuracy : 0.9353
                 95% CI : (0.8872, 0.9673)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8581
 Mcnemar's Test P-Value : 0.07044

            Sensitivity : 0.8571
            Specificity : 0.9813
         Pos Pred Value : 0.9643
         Neg Pred Value : 0.9211
             Prevalence : 0.3706
         Detection Rate : 0.3176
   Detection Prevalence : 0.3294
      Balanced Accuracy : 0.9192

       'Positive' Class : M
```

KNN

Let's try KNN model

```
model_knn <- train(diagnosis~.,
                   data = train_set,
                   method="knn",
                   metric="ROC",
                   preProcess = c('center', 'scale'),
                   tuneLength=10,
                   trControl=fitControl)
```

```
pred_knn <- predict(model_knn, test_set)
cm_knn <- confusionMatrix(pred_knn, test_set$diagnosis, positive = "M")
cm_knn
```

```
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 106  11
         M   1  52

               Accuracy : 0.9294
                 95% CI : (0.8799, 0.963)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8436
 Mcnemar's Test P-Value : 0.009375

            Sensitivity : 0.8254
            Specificity : 0.9907
         Pos Pred Value : 0.9811
         Neg Pred Value : 0.9060
             Prevalence : 0.3706
         Detection Rate : 0.3059
   Detection Prevalence : 0.3118
      Balanced Accuracy : 0.9080

       'Positive' Class : M
```

Neural Networks (NNET)

```
model_nnet <- train(diagnosis~.,
                    data = train_set,
                    method="nnet",
                    metric="ROC",
                    preProcess=c('center', 'scale'),
                    trace=FALSE,
                    tuneLength=10,
                    trControl=fitControl)
```

```
pred_nnet <- predict(model_nnet, test_set)
cm_nnet <- confusionMatrix(pred_nnet, test_set$diagnosis, positive = "M")
cm_nnet
```

```
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 103   3
         M   4  60

               Accuracy : 0.9588
                 95% CI : (0.917, 0.9833)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.912
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9524
            Specificity : 0.9626
         Pos Pred Value : 0.9375
         Neg Pred Value : 0.9717
             Prevalence : 0.3706
         Detection Rate : 0.3529
   Detection Prevalence : 0.3765
      Balanced Accuracy : 0.9575

       'Positive' Class : M
```

Neural Networks (NNET) with PCA

```
model_pca_nnet <- train(diagnosis~.,
                    data = train_set,
                    method="nnet",
                    metric="ROC",
                    preProcess=c('center', 'scale', 'pca'),
                    tuneLength=10,
                    trace=FALSE,
                    trControl=fitControl)
```

```
pred_pca_nnet <- predict(model_pca_nnet, test_set)
cm_pca_nnet <- confusionMatrix(pred_pca_nnet, test_set$diagnosis, positive = "M")
cm_pca_nnet
```

```
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 104   1
         M   3  62

               Accuracy : 0.9765
                 95% CI : (0.9409, 0.9936)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9499
 Mcnemar's Test P-Value : 0.6171

            Sensitivity : 0.9841
            Specificity : 0.9720
         Pos Pred Value : 0.9538
         Neg Pred Value : 0.9905
             Prevalence : 0.3706
         Detection Rate : 0.3647
   Detection Prevalence : 0.3824
      Balanced Accuracy : 0.9780

       'Positive' Class : M
```

SVM with radial kernel

```
model_svm <- train(diagnosis~.,
                   data = train_set,
                   method="svmRadial",
                   metric="ROC",
                   preProcess=c('center', 'scale'),
                   trace=FALSE,
                   trControl=fitControl)
```

```
pred_svm <- predict(model_svm, test_set)
cm_svm <- confusionMatrix(pred_svm, test_set$diagnosis, positive = "M")
cm_svm
```

```
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 101   6
         M   6  57

               Accuracy : 0.9294
                 95% CI : (0.8799, 0.963)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8487
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9048
            Specificity : 0.9439
         Pos Pred Value : 0.9048
         Neg Pred Value : 0.9439
             Prevalence : 0.3706
         Detection Rate : 0.3353
   Detection Prevalence : 0.3706
      Balanced Accuracy : 0.9243

       'Positive' Class : M
```

Naive Bayes

```
model_nb <- train(diagnosis~.,
                  data = train_set,
                  method="nb",
                  metric="ROC",
                  preProcess=c('center', 'scale'),
                  trace=FALSE,
                  trControl=fitControl)
```

```
pred_nb <- predict(model_nb, test_set)
cm_nb <- confusionMatrix(pred_nb, test_set$diagnosis, positive = "M")
cm_nb
```

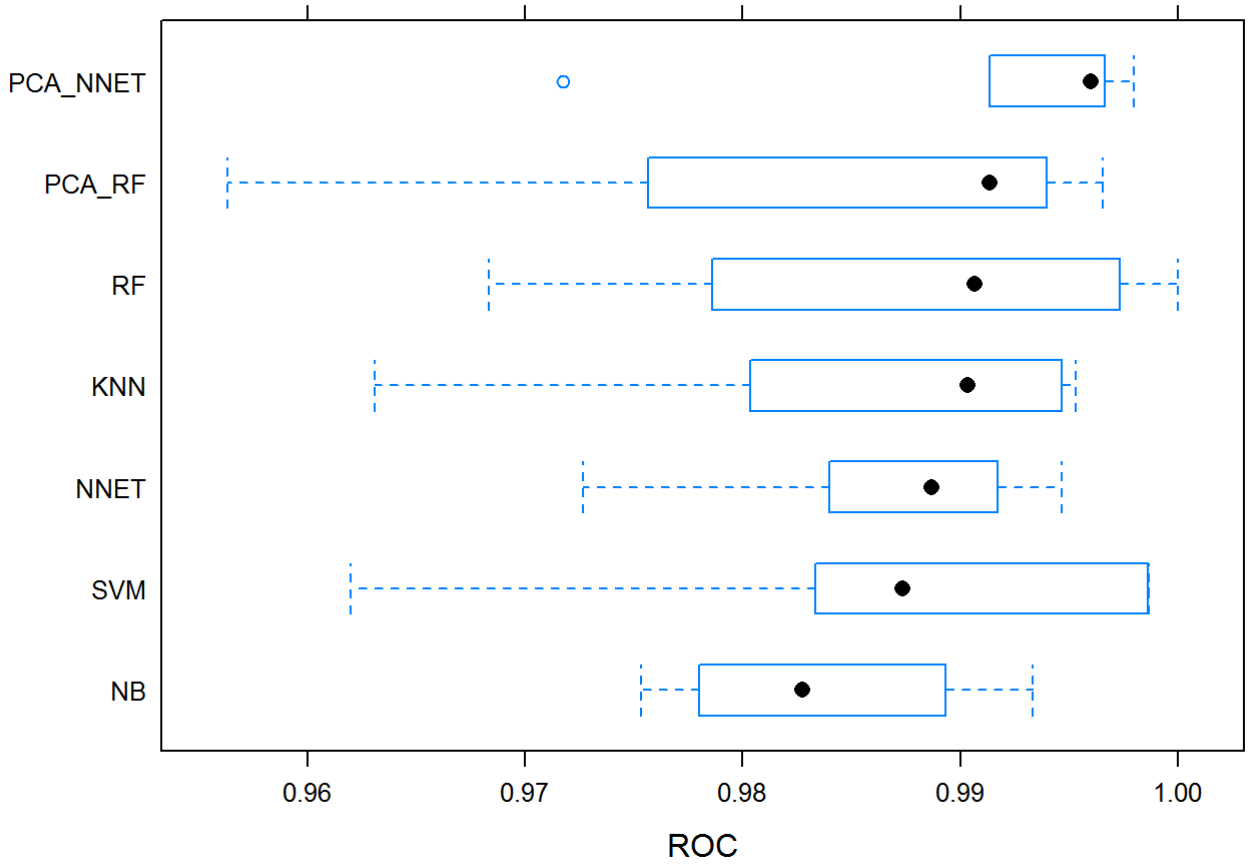
```
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 100   8
         M   7  55

               Accuracy : 0.9118
                 95% CI : (0.8586, 0.9498)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8102
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8730
            Specificity : 0.9346
         Pos Pred Value : 0.8871
         Neg Pred Value : 0.9259
             Prevalence : 0.3706
         Detection Rate : 0.3235
   Detection Prevalence : 0.3647
      Balanced Accuracy : 0.9038

       'Positive' Class : M
```

Models evaluation

Let's compare the models:

```
model_list <- list(RF=model_rf, PCA_RF=model_pca_rf,
                   NNET=model_nnet, PCA_NNET=model_pca_nnet,
                   KNN = model_knn, SVM=model_svm, NB=model_nb)
resamples <- resamples(model_list)
```

```
bwplot(resamples, metric = "ROC")
```



The ROC metric measure the auc of the roc curve of each model. This metric is independent of any threshold.

We see here that some models have a great variability (PCA_RF,RF). The model PCA_NNET achieve a great auc with some variability.

Let's remember how these models result with the testing dataset:

```
cm_list <- list(RF=cm_rf, PCA_RF=cm_pca_rf,
                  NNET=cm_nnet, PCA_NNET=cm_pca_nnet,
                  KNN = cm_knn, SVM=cm_svm, NB=cm_nb)

results <- sapply(cm_list, function(x) x$byClass)
results
```

```
                           RF    PCA_RF      NNET  PCA_NNET       KNN
Sensitivity          0.9047619 0.8571429 0.9523810 0.9841270 0.8253968
Specificity          0.9626168 0.9813084 0.9626168 0.9719626 0.9906542
Pos Pred Value       0.9344262 0.9642857 0.9375000 0.9538462 0.9811321
Neg Pred Value       0.9449541 0.9210526 0.9716981 0.9904762 0.9059829
Precision            0.9344262 0.9642857 0.9375000 0.9538462 0.9811321
Recall               0.9047619 0.8571429 0.9523810 0.9841270 0.8253968
F1                   0.9193548 0.9075630 0.9448819 0.9687500 0.8965517
Prevalence           0.3705882 0.3705882 0.3705882 0.3705882 0.3705882
Detection Rate       0.3352941 0.3176471 0.3529412 0.3647059 0.3058824
Detection Prevalence 0.3588235 0.3294118 0.3764706 0.3823529 0.3117647
Balanced Accuracy    0.9336894 0.9192256 0.9574989 0.9780448 0.9080255
                           SVM        NB
Sensitivity          0.9047619 0.8730159
Specificity          0.9439252 0.9345794
Pos Pred Value       0.9047619 0.8870968
Neg Pred Value       0.9439252 0.9259259
Precision            0.9047619 0.8870968
Recall               0.9047619 0.8730159
F1                   0.9047619 0.8800000
Prevalence           0.3705882 0.3705882
Detection Rate       0.3352941 0.3235294
Detection Prevalence 0.3705882 0.3647059
Balanced Accuracy    0.9243436 0.9037977
```

```
results_max <- apply(results, 1, which.is.max)
```

```
output_report <- data.frame(metric=names(results_max),
                            best_model=colnames(results)[results_max],
                            value=mapply(function(x,y) {results[x,y]},
                            names(results_max),
                                 results_max))
rownames(output_report) <- NULL
output_report
```

```
                 metric best_model     value
1           Sensitivity   PCA_NNET 0.9841270
2           Specificity        KNN 0.9906542
3        Pos Pred Value        KNN 0.9811321
4        Neg Pred Value   PCA_NNET 0.9904762
5             Precision        KNN 0.9811321
6                Recall   PCA_NNET 0.9841270
7                    F1   PCA_NNET 0.9687500
8            Prevalence   PCA_NNET 0.3705882
9        Detection Rate   PCA_NNET 0.3647059
10 Detection Prevalence   PCA_NNET 0.3823529
11    Balanced Accuracy   PCA_NNET 0.9780448
```

The best results for sensitivity (detection of breast cases) is PCA_NNET which also has a great F1 score.

Conclusions

The feature analysis show that there are few features with more predictive value for the diagnosis. The observations were confirmed by the PCA analysis, showing that the same features are aligned to main principal component.

We have found a model based on neural network and PCA preprocessed data with good results over the test set. This model has a sensitivity of 0.984 with a F1 score of 0.968.

```
sessionInfo()
```

```
R version 3.4.1 (2017-06-30)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 8.1 x64 (build 9600)

Matrix products: default

locale:
[1] LC_COLLATE=Hebrew_Israel.1255  LC_CTYPE=Hebrew_Israel.1255
[3] LC_MONETARY=Hebrew_Israel.1255 LC_NUMERIC=C
[5] LC_TIME=Hebrew_Israel.1255

attached base packages:
[1] parallel  grid      stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] doParallel_1.0.11 iterators_1.0.9   foreach_1.4.4
 [4] nnet_7.3-12       purrr_0.2.4       ggfortify_0.4.1
 [7] gridExtra_2.3     pROC_1.10.0       caret_6.0-78
[10] lattice_0.20-35   corrplot_0.84     reshape2_1.4.2
[13] dplyr_0.7.4       e1071_1.6-8       ggplot2_2.2.1

loaded via a namespace (and not attached):
 [1] ddalpha_1.3.1       tidyr_0.7.2         sfsmisc_1.1-1
 [4] splines_3.4.1       prodlim_1.6.1       assertthat_0.2.0
 [7] stats4_3.4.1        DRR_0.0.2           yaml_2.1.16
[10] robustbase_0.92-8   ipred_0.9-6         backports_1.1.2
[13] glue_1.2.0          digest_0.6.13       RColorBrewer_1.1-2
[16] randomForest_4.6-12 colorspace_1.3-2    recipes_0.1.1
[19] htmltools_0.3.6     Matrix_1.2-10       plyr_1.8.4
[22] psych_1.7.8         klaR_0.6-12         timeDate_3042.101
[25] pkgconfig_2.0.1     CVST_0.2-1          broom_0.4.3
[28] scales_0.5.0        ranger_0.8.0        gower_0.1.2
[31] lava_1.5.1          combinat_0.0-8      tibble_1.3.4
[34] withr_2.1.1         lazyeval_0.2.1      mnormt_1.5-5
[37] survival_2.41-3     magrittr_1.5        evaluate_0.10.1
[40] GGally_1.3.2        nlme_3.1-131        MASS_7.3-47
[43] dimRed_0.1.0        foreign_0.8-69      class_7.3-14
[46] tools_3.4.1         stringr_1.2.0       kernlab_0.9-25
[49] munsell_0.4.3       bindrcpp_0.2        compiler_3.4.1
[52] RcppRoll_0.2.2      rlang_0.1.6         labeling_0.3
[55] rmarkdown_1.8       gtable_0.2.0        ModelMetrics_1.1.0
[58] codetools_0.2-15    reshape_0.8.7       DBI_0.7
[61] R6_2.2.2            lubridate_1.7.1     knitr_1.17
[64] bindr_0.1           rprojroot_1.3-1     stringi_1.1.6
[67] Rcpp_0.12.14        rpart_4.1-11        DEoptimR_1.0-8
[70] tidyselect_0.2.3
```