Predicting the Occurrence of Cancer

Caio di Felice Cunha

Definition of the Business Problem

Predicting the Occurrence of Breast Cancer

Data Source: http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

Stage 1 - Collecting the Data

Breast cancer data include 569 observations of cancer biopsies, each with 32 characteristics (variables). A feature is a number of identification (ID), another is cancer diagnosis, and 30 are laboratory measurements numeric. The diagnosis is coded as "M" to indicate malignant or "B" to indicate indicate benign.

```
data <- read.csv("dataset.csv", stringsAsFactors = FALSE)
str(data)
```

```
## 'data.frame':    569 obs. of  32 variables:
##  $ id               : int  87139402 8910251 905520 868871 9012568 906539 925291 87880 862989 89827 .
##  $ diagnosis        : chr  "B" "B" "B" "B" ...
##  $ radius_mean      : num  12.3 10.6 11 11.3 15.2 ...
##  $ texture_mean     : num  12.4 18.9 16.8 13.4 13.2 ...
##  $ perimeter_mean   : num  78.8 69.3 70.9 73 97.7 ...
##  $ area_mean        : num  464 346 373 385 712 ...
##  $ smoothness_mean  : num  0.1028 0.0969 0.1077 0.1164 0.0796 ...
##  $ compactness_mean : num  0.0698 0.1147 0.078 0.1136 0.0693 ...
##  $ concavity_mean   : num  0.0399 0.0639 0.0305 0.0464 0.0339 ...
##  $ points_mean      : num  0.037 0.0264 0.0248 0.048 0.0266 ...
##  $ symmetry_mean    : num  0.196 0.192 0.171 0.177 0.172 ...
##  $ dimension_mean   : num  0.0595 0.0649 0.0634 0.0607 0.0554 ...
##  $ radius_se        : num  0.236 0.451 0.197 0.338 0.178 ...
##  $ texture_se       : num  0.666 1.197 1.387 1.343 0.412 ...
##  $ perimeter_se     : num  1.67 3.43 1.34 1.85 1.34 ...
##  $ area_se          : num  17.4 27.1 13.5 26.3 17.7 ...
##  $ smoothness_se    : num  0.00805 0.00747 0.00516 0.01127 0.00501 ...
##  $ compactness_se   : num  0.0118 0.03581 0.00936 0.03498 0.01485 ...
##  $ concavity_se     : num  0.0168 0.0335 0.0106 0.0219 0.0155 ...
##  $ points_se        : num  0.01241 0.01365 0.00748 0.01965 0.00915 ...
##  $ symmetry_se      : num  0.0192 0.035 0.0172 0.0158 0.0165 ...
##  $ dimension_se     : num  0.00225 0.00332 0.0022 0.00344 0.00177 ...
##  $ radius_worst     : num  13.5 11.9 12.4 11.9 16.2 ...
##  $ texture_worst    : num  15.6 22.9 26.4 15.8 15.7 ...
##  $ perimeter_worst  : num  87 78.3 79.9 76.5 104.5 ...
##  $ area_worst       : num  549 425 471 434 819 ...
##  $ smoothness_worst : num  0.139 0.121 0.137 0.137 0.113 ...
```

```
##  $ compactness_worst: num  0.127 0.252 0.148 0.182 0.174 ...
##  $ concavity_worst  : num  0.1242 0.1916 0.1067 0.0867 0.1362 ...
##  $ points_worst     : num  0.0939 0.0793 0.0743 0.0861 0.0818 ...
##  $ symmetry_worst   : num  0.283 0.294 0.3 0.21 0.249 ...
##  $ dimension_worst  : num  0.0677 0.0759 0.0788 0.0678 0.0677 ...
```

Stage 2 - Pre-Processing

Deleting the ID column Regardless of the machine learning method, it should always be excluded ID variables. Otherwise this can lead to wrong results because the ID can be used to uniquely "predict" each example. Therefore, a model that includes an identifier can suffer from overfitting, and it will be very difficult to use it to generalize other data.

```
data$id = NULL
# Adjusting the label of the target variable
data$diagnosis = sapply(
  data$diagnosis,
  function(x){ifelse(x=='M', 'Malign', 'Benign')})

# Many classifiers require variables to be of type Factor
table(data$diagnosis)
```

```
##
## Benign Malign
##    357    212
```

```
data$diagnosis <- factor(
  data$diagnosis,
  levels = c("Benign", "Malign"),
  labels = c("Benign", "Malign"))

str(data$diagnosis)
```

```
##  Factor w/ 2 levels "Benign","Malign": 1 1 1 1 1 1 1 2 1 1 ...
```

```
# Checking the aspect ratio
round(prop.table(table(data$diagnosis)) * 100, digits = 1)
```

```
##
## Benign Malign
##   62.7   37.3
```

```
# Measures of Central Tendency
# We detected a scaling problem between the data,
# which then needs to be normalized
# The distance calculation done by kNN is dependent on the
# scale measurements in the input data.
summary(data[c("radius_mean", "area_mean", "smoothness_mean")])
```

```
##   radius_mean       area_mean      smoothness_mean
```

```
##  Min.   : 6.981   Min.   : 143.5   Min.   :0.05263
##  1st Qu.:11.700   1st Qu.: 420.3   1st Qu.:0.08637
##  Median :13.370   Median : 551.1   Median :0.09587
##  Mean   :14.127   Mean   : 654.9   Mean   :0.09636
##  3rd Qu.:15.780   3rd Qu.: 782.7   3rd Qu.:0.10530
##  Max.   :28.110   Max.   :2501.0   Max.   :0.16340
```

```
# Creating a normalization function
## Note: This "normalize" function that was created is actually a
## standardization (to standardize the data)

normalizes <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

# Normalizing the data
data_norm <- as.data.frame(lapply(data[2:31], normalizes))
```

Stage 3 - Training the model with KNN

```
# Loading the library package
# install.packages("class")
library(class)
```

```
# Creating training date and test date
sample_df <- sample(c(TRUE, FALSE),
                    nrow(data_norm),
                    replace = TRUE, prob = c(0.7, 0.3))

training_data <- data_norm[sample_df,]
test_data <- data_norm[!sample_df,]

# Creating the labels for the training and test data
training_data_labels <- data[sample_df, 1]
testing_data_labels <- data[!sample_df, 1]
```

```
length(training_data_labels)
```

```
## [1] 410
```

```
length(testing_data_labels)
```

```
## [1] 159
```

```
# Creating the model
model_knn_v1 <- knn(train = training_data,
                    test = test_data,
                    cl = training_data_labels,
                    k = 21)
```

```r
# The knn() function returns a factor object with the predictions for
# each example in the test dataset
summary(model_knn_v1)
```

```
## Benign Malign
##    108     51
```

Stage 4 - Evaluating and Interpreting the KNN model

```r
# Loading gmodels
library(gmodels)

# Creating a cross table of predicted data x current data

CrossTable(x = testing_data_labels, y = model_knn_v1, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  159
##
##
##                      | model_knn_v1
## testing_data_labels |    Benign |    Malign | Row Total |
## --------------------|-----------|-----------|-----------|
##              Benign |        98 |         1 |        99 |
##                     |     0.990 |     0.010 |     0.623 |
##                     |     0.907 |     0.020 |           |
##                     |     0.616 |     0.006 |           |
## --------------------|-----------|-----------|-----------|
##              Malign |        10 |        50 |        60 |
##                     |     0.167 |     0.833 |     0.377 |
##                     |     0.093 |     0.980 |           |
##                     |     0.063 |     0.314 |           |
## --------------------|-----------|-----------|-----------|
##        Column Total |       108 |        51 |       159 |
##                     |     0.679 |     0.321 |           |
## --------------------|-----------|-----------|-----------|
##
##
```

Interpreting the Results The cross table shows 4 possible values, which represent the false/true positive and negative. We have two columns listing the original labels in the observed data. We have two lines listing test data labels

We have:

- Scenario 1: Benign (Observed) x Benign (Predicted) cell - 104 cases - true positive
- Scenario 2: Malign Cell (Observed) x Benign (Predicted) - 00 cases - false positive (the model failed)
- Scenario 3: Benign Cell (Observed) x Malign (Predicted) - 02 cases - false negative (the model failed)
- Scenario 4: Malign Cell (Observed) x Malign (Predicted) - 54 cases - true negative

Reading the Confusion Matrix (Perspective of having or not having the disease):

- True Negative = our model predicted that the person did NOT have the disease and the data showed that the person did NOT have the disease
- False Positive = our model predicted the person had the disease and the data showed NO, the person had the disease
- False Negative = our model predicted that the person did NOT have the disease and the data showed that YES, the person had the disease
- True Positive = our model predicted that the person had the disease and the data showed that YES, the person had the disease

False Positive - Type I Error False Negative - Type II Error

Model hit rate: 98% (hit 98 out of 100)

Stage 5: Optimizing Model Performance

Using the scale() function to standardize the z-score obs: the function of R "scale" is the normalization

```
data_z <- as.data.frame(scale(data[-1]))

# Confirming successful transformation
summary(data_z$area_mean)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.4532 -0.6666 -0.2949  0.0000  0.3632  5.2459
```

```
# Creating new training and testing datasets
sample_df <- sample(c(TRUE, FALSE), nrow(data_norm),replace = TRUE, prob = c(0.7, 0.3))

training_data <- data_z[sample_df,]
test_data <- data_z[!sample_df,]

# Creating the labels for the training and test data
training_data_labels <- data[sample_df, 1]
testing_data_labels <- data[!sample_df, 1]

# Creating the model
model_knn_v2 <- knn(train = training_data,
                    test = test_data,
                    cl = training_data_labels,
```

```
                      k = 21)

# Creating a cross table of predicted date x current date
CrossTable(x = testing_data_labels, y = model_knn_v2, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  174
##
##
##                      | model_knn_v2
## testing_data_labels |    Benign |    Malign | Row Total |
## --------------------|-----------|-----------|-----------|
##              Benign |       107 |         0 |       107 |
##                     |     1.000 |     0.000 |     0.615 |
##                     |     0.930 |     0.000 |           |
##                     |     0.615 |     0.000 |           |
## --------------------|-----------|-----------|-----------|
##              Malign |         8 |        59 |        67 |
##                     |     0.119 |     0.881 |     0.385 |
##                     |     0.070 |     1.000 |           |
##                     |     0.046 |     0.339 |           |
## --------------------|-----------|-----------|-----------|
##        Column Total |       115 |        59 |       174 |
##                     |     0.661 |     0.339 |           |
## --------------------|-----------|-----------|-----------|
##
##
```

```
# Try different values for k
```

Stage 6: Building a model with Support Vector Machine Algorithm (SVM)

```
# Prepare the dataset
data <- read.csv("dataset.csv", stringsAsFactors = FALSE)
data$diagnosis <- factor(data$diagnosis)

data$id = NULL
```

```r
## Create column to separate training and test data
## The creation of this index is random. That is, each time we run this code,
# the index changes, and therefore, the training and test data, too.
data[,'index'] <- ifelse(runif(nrow(data)) < 0.7,1,0)
```

```r
# training and testing date
trainset <- data[data$index==1,]
testset <- data[data$index==0,]

# Get the index
trainColNum <- grep('index', names(trainset))

# Remove index from datasets
trainset <- trainset[,-trainColNum]
testset <- testset[,-trainColNum]

# Get column index of target variable in dataset
## Choose the column that starts with 'diag'
typeColNum <- grep('diag',names(data))

# Create the model
# We set the kernel to radial, as this dataset doesn't have a
# linear plane that can be drawn
library(e1071)

model_svm_v1 <- svm(diagnosis ~ .,
                    data = na.omit(trainset),
                    type = 'C-classification',
                    kernel = 'radial')
```

```r
# Forecasts

# Predictions on training dates
pred_train <- predict(model_svm_v1, trainset)

# Percentage of correct predictions with training dataset
mean(pred_train == trainset$diagnosis)
```

```
## [1] 0.9826303
```

```r
# Forecasts on test date
pred_test <- predict(model_svm_v1, testset)

# Percentage of correct predictions with test dataset
mean(pred_test == testset$diagnosis)
```

```
## [1] 0.9518072
```

```r
# Confusion Matrix
table(pred_test, testset$diagnosis)
```

```
##
```

```
## pred_test  B  M
##         B 94  4
##         M  4 64
```

Step 7: Building a model with Random Forest Algorithm

```
# Creating the model
library(rpart)
model_rf_v1 = rpart(diagnosis ~ .,
                    data = trainset,
                    control = rpart.control(cp = .0005))

# Forecasts on test date
tree_pred = predict(model_rf_v1, testset, type='class')

# Percentage of correct predictions with test dataset
mean(tree_pred==testset$diagnosis)
```

```
## [1] 0.9156627
```

```
# Confusion Matrix
table(tree_pred, testset$diagnosis)
```

```
##
## tree_pred  B  M
##         B 92  8
##         M  6 60
```

Disclaimer:

Disclaimer: a good part of this project was largely done in the Data Science Academy, Big Data Analytics with R and Microsoft Azure Machine Learning course (part of the Data Scientist training)

End