Breast Cancer-Exercises

December 6, 2018

0.1 Exercise

Try with different k choices and do a quick comparison. You can draw a plot to show the results.

0.1.1 Import data file

```
In [1]: # Local directory - use your own!!!!!!
        setwd("/Users/jramongomez/Master/IntroduccionDataScience/CLASIFICACION")
```

```
In [2]: # Load data
        require(caret)
        wbcd <- read.csv("wisc_bc_data.csv", stringsAsFactors = FALSE)
```

```
Loading required package: caret
Loading required package: lattice
Loading required package: ggplot2
```

0.1.2 Preprocess data

```
In [3]: # Drop the id feature
        wbcd <- wbcd[,-1]
```

```
In [4]: # Recode diagnosis as a factor
        wbcd$diagnosis <- factor(wbcd$diagnosis, levels = c("B", "M"),
                                 labels = c("Benign", "Malignant"))
```

```
In [5]: # Normalize the wbcd data
        wbcd_n <- as.data.frame(lapply(wbcd[,2:31], scale, center = TRUE, scale = TRUE))
```

0.1.3 Create training and test datasets

```
In [6]: # Create training and test data
        shuffle_ds <- sample(dim(wbcd_n)[1])
        eightypct <- (dim(wbcd_n)[1] * 80) %/% 100
        wbcd_train <- wbcd_n[shuffle_ds[1:eightypct], ]
        wbcd_test <- wbcd_n[shuffle_ds[(eightypct+1):dim(wbcd_n)[1]], ]

        # Create labels for training and test data
        wbcd_train_labels <- wbcd[shuffle_ds[1:eightypct], 1]
        wbcd_test_labels <- wbcd[shuffle_ds[(eightypct+1):dim(wbcd_n)[1]], 1]
```

0.1.4 Training a model on the data - By hand...

In [7]:
```
# Load the "class" library
library(class)
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test,
                      cl = wbcd_train_labels, k=21)
str(wbcd_test_pred)
```

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

In [8]:
```
# Probamos a calcular el knn usando un tamaño de k como máximo raíz de n, para evitar so
# probando sólo con los valores impares.
#Los valores de K serán los siguientes
vectorK = seq(1,sqrt(dim(wbcd)[1]),2)
knnModel.k <- train(wbcd_train, wbcd_train_labels, method="knn",
                    metric="Accuracy", tuneGrid = data.frame(.k=vectorK))
```

In [9]:
```
knnModel.k
```

```
k-Nearest Neighbors

455 samples
 30 predictor
  2 classes: 'Benign', 'Malignant'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 455, 455, 455, 455, 455, 455, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.9468916  0.8846372
   3  0.9454064  0.8815346
   5  0.9477755  0.8862038
   7  0.9545372  0.9007387
   9  0.9514943  0.8940715
  11  0.9523855  0.8957831
  13  0.9545535  0.9004230
  15  0.9536514  0.8982964
  17  0.9529155  0.8965870
  19  0.9532927  0.8973790
  21  0.9546875  0.9003986
  23  0.9526099  0.8957964

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 21.
```

In [10]:
```
kResults = knnModel.k$results[1:dim(knnModel.k$results)[1],][1:3]
```

In [11]:
```
#En primer lugar me quedaré con los 3 modelos de k cuya precisión sea mayor
primerosK = kResults[order(kResults$Accuracy, decreasing = TRUE)[1:3],]
primerosK
```

	k	Accuracy	Kappa
11	21	0.9546875	0.9003986
7	13	0.9545535	0.9004230
4	7	0.9545372	0.9007387

In [12]:
```
#Calculamos los modelos para los distintos valores de K
knnModel.k1 <- train(wbcd_train, wbcd_train_labels, method="knn",
                     metric="Accuracy", tuneGrid = data.frame(.k=primerosK$k[1]))
knnModel.k2 <- train(wbcd_train, wbcd_train_labels, method="knn",
                     metric="Accuracy", tuneGrid = data.frame(.k=primerosK$k[2]))
knnModel.k3 <- train(wbcd_train, wbcd_train_labels, method="knn",
                     metric="Accuracy", tuneGrid = data.frame(.k=primerosK$k[3]))
```

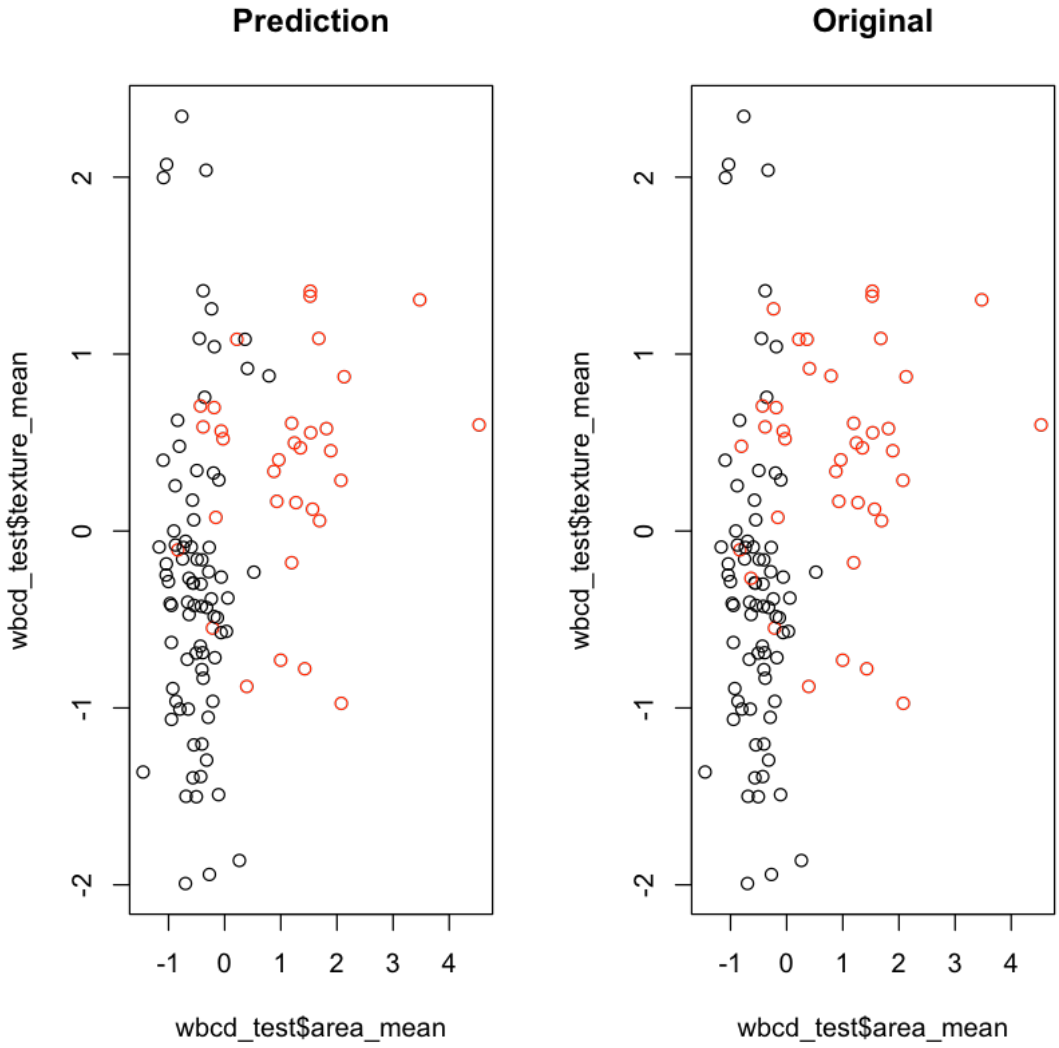
In [13]:
```
# A continuación calcularemos los errores en las predicciones

knnPred.k1 <- predict(knnModel.k1, newdata = wbcd_test)
postResample(pred = knnPred.k1, obs = wbcd_test_labels)

knnPred.k2 <- predict(knnModel.k2, newdata = wbcd_test)
postResample(pred = knnPred.k2, obs = wbcd_test_labels)

knnPred.k3 <- predict(knnModel.k3, newdata = wbcd_test)
postResample(pred = knnPred.k3, obs = wbcd_test_labels)
```

Accuracy 0.947368421052632 **Kappa** 0.878594249201278
Accuracy 0.973684210526316 **Kappa** 0.940438871473354
Accuracy 0.982456140350877 **Kappa** 0.960539979231568

Como hemos podido apreciar, las medidas de precisión (Accuracy) y Kappa eran muy similares para los valores de k que hemos escogido. Al predecir las etiquetas para nuestro conjunto test utilizando dichos clasificadores knn hemos observado que la precisión y Kappa (mide cuanto mejor o peor es un clasificador que una asignación aleatoria) con respecto al test son iguales, así que sólo hemos representado una de ellas y también hemos pintado los datos originales.

También cabe destacar que hemos elegido las variables texture_mean y area_mean para realizar el gráfico debido a que por lo que hemos observado anteriormente en las gráficas y las medidas de correlación, son las que más nos podían ayudar para realizar una buena clasificación.

In [14]:
```
par(mfrow=c(1,2))
plot(wbcd_test$texture_mean~wbcd_test$area_mean,col=knnPred.k1, main = "Prediction")
plot(wbcd_test$texture_mean~wbcd_test$area_mean,col=wbcd_test_labels, main = "Original"
```



Como podemos apreciar apenas existe diferencia, las diferencias radican en datos muy cercanos.