Assignment 1 – k-NN and Logistic Regression

Jonathan De Los Santos

2/2/2021

# Question 1

“Explore and prepare data by using the str() function. Display the probability of the attributes (‘benign’ and ‘malignant’) of the variable named “diagnosis” that we plan to use for prediction and normailize the entire dataset. (20 pts)"

Begin by loading the Wisconsin Breast Cancer Data Set. Exploring the structure, we notice and unnecessary ID column that we drop from the data frame.

bcdata <- read.csv("Data Sets/1.2-wisc\_bc\_data.csv")  
  
# Drop the id column, it won't be needed  
bcdata <- bcdata[-1]  
  
#str(bcdata)

To determine the probability of variables “benign” and “malignant” (coded here as “B” and “M”), we create a table of the diagnosis values proportions then convert it to percent. Here we see that the probability of value benign is 62.7% while the probability of malignant is 37.3%.

round(prop.table(table(bcdata$diagnosis))\*100, digits = 1)

##   
## B M   
## 62.7 37.3

### Data Prep

Here we create a function to perform a normalization, apply that function over a vector of all columns *except* the diagnostic column, and save it in a new data frame as normalized data. A head() of the data shows the result.

# Create normalization function  
normalize <- function(x){  
 return ((x-min(x))/(max(x)-min(x)))  
   
bcdata\_n <- bcdata  
   
# Apply normalization function over the data, cast it to new data frame   
bcdata\_n[, 2:31] <- as.data.frame(lapply(bcdata[2:31], normalize))  
  
#str(bcdata\_n)  
}

# Question 2

“Create datasets for training and testing the model, and develop the model using the knn classifier algorithm. Evaluate the model with different k, and propose the best value of k.”

## 2-A

“Split the dataset into training and testing. The proportions of training and testing dataset will be 7:3. (20 pts)”

To split the data sets into training and testing, we create a training index of 70% of the data set rows, and a test index of the difference. These are stored in their respective dataframes.

library(caret)  
train.index <- sample(row.names(bcdata\_n), 0.7\*dim(bcdata\_n)[1])  
test.index <- setdiff(row.names(bcdata\_n), train.index)  
  
train.df <- bcdata\_n[train.index, ]  
test.df <- bcdata\_n[test.index, ]

## 2-B

“Develop the model using the knn classifier algorithm. (20 pts)”

We run the knn function with the train and test data using the first column as our classifier.

library(class)  
knn(train = train.df[, 2:31], test = test.df[,2:31],  
 cl = train.df[,1], k = 3)

## [1] M B M M M M M B B M M M B M B B B B M B M B M B M B B B B B M B M B B B B M M B M B M M B B B  
## [48] B B M B B B B M B B B B B B M B M B B M M M M B M M B M B B B B B M B B B M M M M B B B B B B  
## [95] B B B B B B M B M B M B M B B B B B B M M B B B B M B B B B B M B M B B B B B B M B B B M B B  
## [142] B B M B B B B B B B B M M B B B B B B B B B B M M B B B B M  
## Levels: B M

## 2-C

“Evaluate the model’s performance for different K, and suggest the best model. (20 pts)”

To evaluate the model, we create a dataframe for storing the accuracy of the model over several values of . We loop over the execution of the model storing the results in accuracy.df. Based on the results, it appears that k = 10, 11 give the highest accuracy rates. We should choose 10 as it would be slightly more performant.

# Initialize a data frame with columns k and accuracy  
accuracy.df <- data.frame (k = seq(1, 15, 1), accuracy = rep(0,15))  
  
# Calculate range of k-values to see which validate the best   
# Loop i-k  
for (i in 1:15) {  
 knn.pred <- knn(train.df[, 2:31], test.df[, 2:31],  
 cl = train.df[, 1], k = i)  
 accuracy.df[i, 2] <- confusionMatrix(knn.pred, factor(test.df[, 1]))$overall[1]  
}  
  
print(accuracy.df)

## k accuracy  
## 1 1 0.9649123  
## 2 2 0.9649123  
## 3 3 0.9766082  
## 4 4 0.9766082  
## 5 5 0.9766082  
## 6 6 0.9766082  
## 7 7 0.9883041  
## 8 8 0.9883041  
## 9 9 0.9883041  
## 10 10 0.9883041  
## 11 11 0.9883041  
## 12 12 0.9941520  
## 13 13 0.9941520  
## 14 14 0.9883041  
## 15 15 0.9883041

#str(test.df)  
#str(knn.pred)  
  
knn.best <- knn(train = train.df[, 2:31], test = test.df[,2:31],  
 cl = train.df[,1], k = 10)  
  
#str(knn.best)

# Question 3

“Build a logistic regression model with the same dataset above and the data partition to develop the best possible diagnostic machine learning algorithm to assist the medical team in determining whether the tumor is malignant or not. In this analysis, you need to standardize the data, and develop a logistic regression model. Provide a detailed explanation of the output and the comparison with the outcomes of knn in Question 2. (20 pts)”

To begin, I chose to map the diagnosis as a factor, then as a character with labels, and finally back to numeric resulting in a value of 1 for “M” and 0 for “B”

dataset<- read.csv("Data Sets/1.2-wisc\_bc\_data.csv")  
dataset$diagnosis.r <- as.numeric(as.character(factor(dataset$diagnosis, c("M", "B"), labels = c(1, 0))))  
#head(dataset, 20)

Partitioned data if needed for testing:

partition <- createDataPartition(y = dataset$diagnosis.r, p = 0.5, list = FALSE)  
train <- dataset[partition,]  
test <- dataset[-partition,]  
#print(test)

I’m not entirely sure of the the direction to develop the “best” algorithm, but to demonstrate an example I chose the variables “texture mean” and “perimiter mean”

diagnostic\_model <- glm(diagnosis.r~texture\_mean+perimeter\_mean, family = "binomial", data = dataset)  
diagnostic\_model

##   
## Call: glm(formula = diagnosis.r ~ texture\_mean + perimeter\_mean, family = "binomial",   
## data = dataset)  
##   
## Coefficients:  
## (Intercept) texture\_mean perimeter\_mean   
## -20.4164 0.2224 0.1677   
##   
## Degrees of Freedom: 568 Total (i.e. Null); 566 Residual  
## Null Deviance: 751.4   
## Residual Deviance: 268.1 AIC: 274.1

To get odds we plug this into an exponential function:

exp(coef(diagnostic\_model))

## (Intercept) texture\_mean perimeter\_mean   
## 1.359165e-09 1.249098e+00 1.182628e+00

Here we find that the odds of the diagnosis being malignant increase by 1 for every 1-unit increase in texture\_mean and perimeter\_mean. Units are not given for the perimeter, but texture is understood to be the standard deviation of gray-scale values. This implies that cells with higher differences in their image have a higher odds of being malignant.