Week 8 Exercise 14

Nosky, Christopher

2/19/2021

Include all of your answers in a R Markdown report. Here is an example R Markdown report that you can use as a guide.

Fit a logistic regression model to the binary-classifier-data.csv dataset from the previous assignment.

LogModel.1 <- glm(label~x+y, data = binary\_train, family = binomial())  
summary(LogModel.1)

##   
## Call:  
## glm(formula = label ~ x + y, family = binomial(), data = binary\_train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.3658 -1.1672 -0.9614 1.1650 1.4004   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.415200 0.143553 2.892 0.003824 \*\*   
## x -0.002376 0.002237 -1.062 0.288036   
## y -0.007953 0.002302 -3.456 0.000549 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1382.9 on 997 degrees of freedom  
## Residual deviance: 1367.6 on 995 degrees of freedom  
## AIC: 1373.6  
##   
## Number of Fisher Scoring iterations: 4

1. What is the accuracy of the logistic regression classifier?

LogModel.1.train <- predict(LogModel.1, binary\_train, type= 'response')   
  
LogModel.1.test <- predict(LogModel.1, binary\_test, type = 'response')  
  
# Model 1 Train confusion matrix  
confmatrix.1 <- table(Actual\_Value=binary\_train$label, Predicted\_Value = LogModel.1.train > 0.5)  
confmatrix.1

## Predicted\_Value  
## Actual\_Value FALSE TRUE  
## 0 289 222  
## 1 202 285

# Model 1 Test confusion matrix  
confmatrix.2 <- table(Actual\_Value=binary\_test$label, Predicted\_Value = LogModel.1.test > 0.5)  
confmatrix.2

## Predicted\_Value  
## Actual\_Value FALSE TRUE  
## 0 143 113  
## 1 98 146

# Model 1 Train acc  
(confmatrix.1[[1,1]] + confmatrix.1[[2,2]]) / sum(confmatrix.1)

## [1] 0.5751503

# Model 1 Test acc  
(confmatrix.2[[1,1]] + confmatrix.2[[2,2]]) / sum(confmatrix.2)

## [1] 0.578

1. How does the accuracy of the logistic regression classifier compare to the nearest neighbors algorithm?

* First we’ll split the data into training and test sets.

# Split for training / test  
  
set.seed(123) #Getting random sample  
dat.d <- sample(1:nrow(binary\_data), size=nrow(binary\_data)\*0.8, replace = FALSE) # Selects 80% of data at random  
  
train.binary <- binary\_data[dat.d,] # 80% training data  
test.binary <- binary\_data[-dat.d,] # remaining 20% for testing  
  
# Now creating seperate dataframe.  
train.binary\_labs <- binary\_data[dat.d,1]  
test.binary\_labs <- binary\_data[-dat.d, 1]  
  
NROW(train.binary\_labs)

## [1] 1198

* Next we’ll build our Knn model where K=3. As you can see the accuracy comes in at 99.3333%, which is confirmed with the confusion matrix.

bin\_knn\_3 <- knn(train=train.binary, test=test.binary, cl=train.binary\_labs, k=3)  
  
# Let's calculate the proprotion of correct classification for k = 3  
  
bin\_acc\_3 <- 100 \* sum(test.binary\_labs == bin\_knn\_3)/NROW(test.binary\_labs) # For knn = 3  
bin\_acc\_3 # Accuracy

## [1] 99.33333

table(bin\_knn\_3, test.binary\_labs) # to check prediction against actual value in tabular form

## test.binary\_labs  
## bin\_knn\_3 0 1  
## 0 158 2  
## 1 0 140

# ConfusionMatrix  
confusionMatrix(table(bin\_knn\_3, test.binary\_labs))

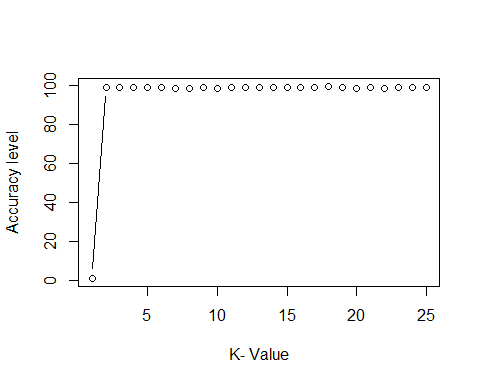
## Confusion Matrix and Statistics  
##   
## test.binary\_labs  
## bin\_knn\_3 0 1  
## 0 158 2  
## 1 0 140  
##   
## Accuracy : 0.9933   
## 95% CI : (0.9761, 0.9992)  
## No Information Rate : 0.5267   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9866   
##   
## Mcnemar's Test P-Value : 0.4795   
##   
## Sensitivity : 1.0000   
## Specificity : 0.9859   
## Pos Pred Value : 0.9875   
## Neg Pred Value : 1.0000   
## Prevalence : 0.5267   
## Detection Rate : 0.5267   
## Detection Prevalence : 0.5333   
## Balanced Accuracy : 0.9930   
##   
## 'Positive' Class : 0   
##

* Next lets look at different values of K, to see if we can get a better fit. As you can see it appears that we may be able to get a high degree of accuraccy from all of our tested values with K = 18 coming in at 99.66667%

i=1 # Declaration to initiate for loop  
bin.k.optm=1 # Declaration to initiate for loop  
for (i in 2:25){  
 bin.knn.mod <- knn(train=train.binary, test=test.binary, cl=train.binary\_labs, k=i)  
 bin.k.optm[i] <- 100 \* sum(test.binary\_labs == bin.knn.mod)/NROW(test.binary\_labs)  
 k=i  
 cat(k,'=', bin.k.optm[i], '\n')  
}

## 2 = 99.33333   
## 3 = 99.33333   
## 4 = 99   
## 5 = 99   
## 6 = 99   
## 7 = 98.66667   
## 8 = 98.66667   
## 9 = 99   
## 10 = 98.66667   
## 11 = 99.33333   
## 12 = 99.33333   
## 13 = 99.33333   
## 14 = 99.33333   
## 15 = 99.33333   
## 16 = 99   
## 17 = 99.33333   
## 18 = 99.66667   
## 19 = 99   
## 20 = 98.66667   
## 21 = 99   
## 22 = 98.66667   
## 23 = 99   
## 24 = 99   
## 25 = 99.33333

plot(bin.k.optm, type='b', xlab='K- Value', ylab='Accuracy level')



1. Why is the accuracy of the logistic regression classifier different from that of the nearest neighbors?

* There are likely a number of reasons that the accuracy is different between the two methods. As the reading points out KNN is a lazy execution, it fits and predicts at the time it is ran. KNN doesn’t tell us which of our predictors are significant, where logistic regression we can approach it in a backwards stepwise method and see the coeficients and pvalues of our predictors to find the variables with the highest significance.