INTELIGENCIA DE NEGOCIOS: TAREA 3

Sergio Miranda

Francisco Merino

Alex Escudero

28 Octubre 2017

En el presente informe, se desarrollará la tercera tarea del semestre.

Descripción:

Una compañía de servicios está recibiendo diariamente una gran cantidad de e-mail del tipo SPAM. Cansado de esta situación quiere ver la posibilidad de crear un modelo de clasificación para descartar este tipo de correos. Para ello, la compañía recolectó un total de 700 e-mails en el archivo features-train.txt. Este archivo consta de 3 columnas, la primera columna es el ID del e-mail (de 1 a 700), la segunda columna corresponde a un número que se asocia a una palabra en particular, y la tercera columna corresponde al número de veces que esa palabra fue utilizada en el correo electrónico. Por ejemplo, el segundo correo (ID = 2) fue resumido en 5 palabras, de las cuales la palabra 166 (area) fue utilizada 1 vez. La clase de cada e-mail se encuentra en el archivo train-labels.txt. Este archivo de valores binarios (0 o 1), tiene valor 0 si es un e-mail verdadero, mientras que un valor de 1 implica spam. Finalmente, el listado de palabras se encuentra en el archivo listFeatures.txt. El propósito de la tarea es entrenar modelos predictivos sobre estos datos y evaluarlos sobre 260 nuevos e-mails. Para esto, se le solicita que Ud. tome los datos de la compañía y realice programas en R que respondan las siguientes preguntas:

1) Aplique el modelo KNN sobre los datos originales utilizando distintos valores de K y medidas de distancia. Para esto, deberá utilizar el modelo. Además, aplique el modelo modificando el parámetro K desde 1 hasta 20. Para cada aplicación del modelo calcule el porcentaje de e-mails de test bien clasificados (ver las características y el tipo de e-mail en features-test.txt y test-labels.txt, respectivamente). Luego elija el mejor modelo indicando el valor para KNN y la medida de distancia.

Para el desarrollo de esta primera pregunta, se realizó un ajuste de los Dataframes, para así poder ser procesados en R. Esto dado a que las muestras eran distintas y la distribución de sus variables no coincidían.

Mediante el siguiente script, se cambia la disposición de los datos, dejando en cada fila un email y en cada columna, una palabra.

```
#Importamos las tablas
setwd("/Users/FcoMerino/Documents/UNIVERSIDAD/inteligencia\ de\
negocios")
```

```
emails = read.table("features-train.txt")
emails_class = read.table("train-labels.txt")
emails2 = read.table("features-test.txt")
emails2_class = read.table("test-labels.txt")
#De querer agregar una columna id a emails_class
#emails_class$ID <- seq.int(nrow(emails_class))
#Reshape dataframe
new_emails<-spread(emails, key = V2, value = V3)
new_emails2<-spread(emails2, key = V2, value = V3)
#Llenamos de 0 los NA
new_emails[is.na(new_emails)]<-0
new_emails2[is.na(new_emails2)]<-0
```

A continuación guardamos los datos como matrices según su nombre (Train y Test). A su vez, las generamos dos matrices más con carácter de plantilla, respetando la cantidad de observaciones que cada una contiene (train=700 y test=260).

```
#Guardamos los datos
train_emails<-new_emails
test_emails<-new_emails2

#plantilla para test
plantilla_test<-matrix(data= NA, nrow = 260, ncol=2867, byrow=FALSE)
plantilla_test<-as.data.frame(plantilla_test)
plantilla_test[is.na(plantilla_test)]<-0
plantilla_test<-cbind(X1=1:260, plantilla_test)
colnames(plantilla_test) <- c("V1", c(1:2866))

#plantilla para train
plantilla_train<-matrix(data= NA, nrow = 700, ncol=2867, byrow=FALSE)
plantilla_train<-as.data.frame(plantilla_train)
plantilla_train[is.na(plantilla_train)]<-0
plantilla_train<-cbind(X1=1:700, plantilla_train)
colnames(plantilla_test) <- c("V1", c(1:2866))
```

A continuación, aplicamos un ciclo “FOR” para rellenar las plantillas anteriores, con los datos contenidos en las matrices train_emails y test_emails. A demás agregamos la clase a la que pertenece cada email de los datos.

```
#ciclo para test
for(i in 1:260){
  for(j in 1:2740){
    if ((test_emails[i,j]!=0) && plantilla_test$V1[i]==test_emails$V1[i]){
      plantilla_test[i,j] <- test_emails[i,j]
    } else {
      plantilla_test[i,j] <- 0
    }
```

```
  }
  cat("loop: " , i, "\n")
}


#ciclo para train
for(i in 1:700){
  for(j in 1:2862){
    if ((train_emails[i,j]!=0) && plantilla_train$V1[i]==train_emails$V1[i]){
      plantilla_train[i,j] <- train_emails[i,j]
    } else {
      plantilla_train[i,j] <- 0
    }
  }
  #CAT NOS MUESTRA EL AVANCE EN LA CONSOLA
  cat("loop: " , i, "\n")
}

#Agrego columna clase a datframe final
plantilla_train["Class"]<-emails_class$V1
plantilla_test["Class"]<-emails2_class$V1
```

Por último, guardamos los datos en otra variable y los son exportados a un archivo.

```
#GUARDAMOS
test_knn<-plantilla_test
train_knn<-plantilla_train
write.table(test_knn,
"/Users/FcoMerino/Documents/UNIVERSIDAD/inteligencia\ de\
negocios/test_knn.txt", sep="\t")
write.table(train_knn,
"/Users/FcoMerino/Documents/UNIVERSIDAD/inteligencia\ de\
negocios/train_knn.txt", sep="\t")
```

A partir de este punto, ya podemos aplicar correctamente el algoritmo de KNN, para esto llamamos las tablas necesarias, las librerías y a demás calculamos para cada “k”, su respectiva Accuaracy.

```
library(class)
library(e1071)
library(gmodels)
library(tidyr)

setwd("/Users/FcoMerino/Documents/UNIVERSIDAD/inteligencia\ de\
negocios")
train_knn = read.table("train_knn.txt")
test_knn = read.table("test_knn.txt")
```

```
#KNN=1
emails_knn1<-knn(train_knn[,-c(1,2869)], test_knn[,-
c(1,2869)],as.factor(train_knn[,2869]),k=1)
z<-table(test_knn[,2869],emails_knn1)
a1=((z[1,1]+z[2,2])/(z[1,2]+z[2,1]+z[1,1]+z[2,2]))
a1

#Por razón de espacio, se presenta primer y último k, el resto es visible en script
R

#KNN=20
emails_knn1<-knn(train_knn[,-c(1,2869)], test_knn[,-
c(1,2869)],as.factor(train_knn[,2869]),k=20)
z<-table(test_knn[,2869],emails_knn1)
a20=((z[1,1]+z[2,2])/(z[1,2]+z[2,1]+z[1,1]+z[2,2]))
a20
```

De estos valores, nos quedamos con el K de mayor predicción (53%), en este caso, k=14 .

2) Aplique el modelo de Naive Bayes sobre los datos originales y calcule el error en los datos de test. Para esto, deberá entrenar el modelo Naive Bayes utilizando los 700 e–mails de entrenamiento con y sin la corrección de Laplace. Elija el mejor de estos dos modelos, evaluando el número de e-mails clasificados correctamente.

A partir de este punto, se utilizan las matrices generadas anteriormente y se aplica el algoritmo directamente. Obtenemos a su vez, cada nivel de Accuracy:

```
#Naive Bayes classification without data discretization
bayes <- naiveBayes(Class~.,data=train_knn[,-1])
prediccion <- predict(bayes,test_knn[,-c(1,2869)],type="raw")
pred1 <- max.col(prediccion)
z<-table(test_knn$Class,t(pred1))
```

```
Accuaracy=((z[1,1]+z[2,2])/(z[1,2]+z[2,1]+z[1,1]+z[2,2]))
Accuaracy

#Naive Bayes classification without data discretization (LaPlace)
bayes <- naiveBayes(Class~.,data=train_knn[,-1], Laplace=1)
prediccion <- predict(bayes,test_knn[,-c(1,2869)],type="raw")
pred1 <- max.col(prediccion)
z<-table(test_knn$Class,t(pred1))
Accuaracy=((z[1,1]+z[2,2])/(z[1,2]+z[2,1]+z[1,1]+z[2,2]))
Accuaracy
```

Siendo para el primer proceso 0.4884615 y para Laplace el mismo valor.

3) ¿Qué modelo seleccionaría entre KNN y Naive Bayes cuando utiliza todos las variables? Para esto, compare los errores de los modelos seleccionados en las preguntas 2 y 3, y determine qué modelo produce un error más pequeño.

Claramente, nos quedamos con el modelo KNN, ya que este nos da una mayor asertividad :

KNN(K=14)->53% V/S Naive Bayes->49%

4) En muchos problemas, considerar todas las variables existentes pueden causar ruido en vez de mejorar la clasificación. Elija el modelo de Naive Bayes con corrección de Laplace y busque un subset de 6 variables que mejore los resultados obtenidos en 2. Reporte las variables y su nuevo porcentaje de clasificación correcta. Además, explique el motivo de su nuevo performance. Para ello, entrene un modelo de Naive Bayes con corrección de LaPlace para cada variable (2867 modelos distintos) y seleccione la variable que obtenga el mayor porcentaje de clasificación. Una vez seleccionada la variable, cree 2866 nuevos modelos utilizando la variable previamente seleccionada y cada una de las posibles 2866 variables restantes. Luego elija la mejor variable. Procesa con este mecanismo hasta que obtenga 4 variables distintas.

Para el desarrollo de esta pregunta, es preferente utilizar el peso de cada variable viendo cuales son aquellas 5 que más veces aparecen en la matriz train.

Para esto, utilizamos la siguiente línea de código

```
x<-train_knn[,-c(2869,1)]
tail(sort(colSums(x)),6)
```

Aquí obtenemos las palabras:

```
v2862 v2863 v2864 v2865 v2866 v2867
```

Ahora que tenemos las variables de mayor peso, aplicamos el algoritmo de Naive Bayes para ambas matrices, pero solamente seleccionando las columnas definidas por las variables obtenidas anteriormente.

```
#Matrices con variables seleccionadas
test<-test_knn[,c(1,2206,1704,2583,1353,45,794,2869)]
train<-train_knn[,c(1,2206,1704,2583,1353,45,794,2869)]

#Naive Bayes classification without data discretization (LaPlace)
bayes <- naiveBayes(Class~.,data=train[,-1], Laplace=1)
prediccion <- predict(bayes,test[,-c(1,2869)],type="raw")
pred1 <- max.col(prediccion)
z<-table(test_knn$Class,t(pred1))
Accuaracy=((z[1,1]+z[2,2])/(z[1,2]+z[2,1]+z[1,1]+z[2,2]))
Accuaracy
```

Obteniendo una mejor predicción que aquella realizada con todos los datos de la matriz.

5) ¿Por qué cree las variables del punto 4 fueron seleccionadas?
Básicamente por su frecuencia en las matrices de testeo y entrenamiento. Como se mencionó en el inciso anterior, definimos el "peso" de las variables como la cantidad de presencia de cada una de estas en las matrices, siendo estas las que representan de mejor forma la totalidad de los datos al aplicar algoritmos como Naive Bayes. El hecho de utilizar todas las variables puede ser causa de ruido debido a outliers o una cierta sección de variables con poco peso que no alcanzan a demostrar las tendencias de la matriz, es debido a esto que es bueno hacer un recorte de la cantidad de variables que son usadas para aplicar el algoritmo.

6) Al aplicar un método de selección de variables a KNN usando 9 vecinos y la distancia de Jaccard, se determinó que las 7 mejores variables son: 1041, 2704, 2156, 2595, 452, 1206, 2257. ¿Por qué cree que estas 7 variables fueron seleccionadas?

Primero recordando que la distancia de Jaccard se define como el tamaño de la intersección dividido por el tamaño de la unión entre dos conjuntos (variables para este caso) para una muestra definida. Al tomar el algoritmo KNN con un parámetro K = 9 estamos definiendo las distancias Jaccarianas más cercanas entre los 9 vecinos más parecidos a nivel de conjunto entre ellos.

Si se han determinado las 7 mejores variables aplicando KNN con distancia de Jaccard se quiere decir que estas 7 variables dentro de un rango definido de los 9 vecinos más cercanos, son las más parecidas entre sí a nivel de conjunto, es decir, la intersección de los elementos entre estas variables dividida la totalidad de elementos de ambas variables (su unión), siendo así las variables que mejor representan la cantidad de datos siendo medidos.