UNIVERSIDAD
NACIONAL DEL OESTE

Explotación de Datos

Actividad N^{o} 4

Arboles de Decisión

Profesores:

Dejean, Gustavo
Españadero, Juan
Mendoza, Dante

Integrantes Grupo B:

Benitez, Nicolas
Garcia Ravlic, Ignacio Agustin
Rechimon, Pablo Hernan
Rodriguez, Miguel Angel

Fecha de entrega:

10 de Octubre de 2020

Resumen

A partir de un enorme dataset, provisto por Kaggle, que contiene digitos dibujados a mano en forma de una grilla de pixeles, se planteo el objetivo de desarrollar un modelo capaz de predecir con exactitud el digito dibujado. Para esto utilizamos el algoritmo de Random Forest, aplicandolo en el lenguaje R, obteniendo el mejor modelo con una tasa de aciertos del 96.564%

Palabras Clave:
Árbol de Decision - Análisis de Datos - Ciencia de Datos - Clasificacion - Explotación de datos - Lenguaje R - Modelo - Predicción - Random Forest - Ranger

---

Índice

Graficos

1 Introducción

1.1 Problemática

Se necesita crear un modelo de clasificacion competitivo para predecir el digito dibujado sobre una grilla, la cual esta representada a travez de los 784 pixeles

1.2 Datos a utilizar

Utilizamos los datasets provistos por Kaggle, divididos en *test.csv* y *train.csv*.

1.3 Objetivo

Se busca que el modelo tenga un casi perfecto porcentaje de aciertos para los datos con los que contamos

2 Desarrollo

2.1 Análisis de los datos

Nuestro dataset de entrenamiento, train.csv, contiene 785 variables con 48000 filas. La primera variable, label, contiene el dígito dibujado por el usuario, mientras que el resto contienen los valores en pixel asociados con la imágen correspondiente. El dataset de prueba, test.csv, es idéntico al dataset de entrenamiento, solo que no contiene la variable label.

2.2 Preparación de los datos

Preparamos los datos en base a nuestro requerimientos:

- Comprobación de valores desconocidos.
- Pasamos valores de label a factor.
- Dividimos el dataset train en train y test para pruebas de robustez locales

```
smp_size <- floor(0.80 * nrow(data))
set.seed(123)

train_ind <- sample(seq_len(nrow(data)), size = smp_size)
data.train <-  data[train_ind, ]
data.test <- data[-train_ind, ]
```

Fig. 1: Dividir dataset en entrenamiento y prueba

2.3 Creación y Entrenamiento del Modelo

Antes de comenzar a crear los modelos ejecutamos la función **tuneRF** para encontrar el número óptimo de variables a utilizar en el árbol de descicion.

```
###---------- Important Vars
tuneRF(x = data.train, y = data.train$label,
       mtryStart  = 33, stepFactor = 2,
       ntreeTry   = 600, improve    = 0.01)
# Out:
# mtry = 33   OOB error = 1.32%
# mtry = 66   OOB error = 0.52%
# mtry = 132  OOB error = 0.08%
# mtry = 264  OOB error = 0%
```

Fig. 2: Numero óptimo de Variables

Como resultado obtuvimos que el número minimo óptimo de variables es 33 con un error OOB del 1.32%.

Aplicamos varios random forest a través de la función **ranger()** con diferentes valores de mtry a un primer modelo, comprobando el porcentaje de aciertos del mismo. En este primer modelo se pasa como parámetro TRUE a la variable probability

```
model1 <- ranger( label ~ . , data= data.train[,]
                  , num.trees = 1000
                  , mtry = qvar
                  , importance = "impurity"
                  , write.forest = T
                  , probability = T
                  , alpha = 0.005
) # Out:
model1$prediction.error
# qvars 33 -> OOB: 0.09154588
# qvars 66 -> OOB: 0.08434042
# qvars 132 -> OOB: 0.07927676
```

Fig. 3: Ranger del modelo 1

Podemos observar en los comentarios los resultados del error OOB para un árbol de 33, 66 y 132 variables.

Hicimos un segundo modelo con la variable probability como FALSE para obtener la prediccion en forma de factor y no de probabilidad.

```
model2 <- ranger( label ~ . , data= data.train[,]
                  , num.trees = 1000
                  , mtry = qvar
                  , importance = "impurity"
                  , write.forest = T
                  , probability = F
                  , alpha = 0.005
)
aciertos <- sum(diag(model2$confusion.matrix)) / sum(model2$confusion.matrix) * 100
aciertos
# Salida:
# qvars 33: 96.63393
# qvars 66: 96.61607
# qvars 132: 96.57143
```

Fig. 4: Ranger del modelo 2

En este caso los comentarios nos muestran el porcentaje de aciertos

2.4 Robustez del Modelo

Para comprobar que el modelo es robusto se ejecuto un bucle en el que:

- Se toma una muestra de entrenamiento y otra de prueba

- Se crea un modelo a partir de un numero de variables a tomar predefinido y la muestra de entrenamiento
- Se aplica el modelo a la muestra de prueba
- Se contrasta la prediccion realizada por el modelo con el valor real para obtener el porcentaje de acierto
- Se almacena el resultado en un vector

Al terminar la iteracion se muestra el promedio de acierto entre todas las predicciones

```
vAciertos_RF=c(0)

for(j in 1:10){
  train_ind <- sample(seq_len(nrow(data)), size = smp_size)

  data.train <- data[train_ind, ]
  data.test <- data[-train_ind, ]

  modelo.ranger <- ranger( label ~ . , data= data.train[,]
                           , num.trees = 1000
                           , mtry = qvar
                           , importance = "impurity"
                           , write.forest = T
                           , probability = F
                           , alpha = 0.01
  )

  prediccion <- predict(modelo.ranger, data.test)
  mc <- with(data.test,table(prediccion$predictions, data.test$label))

  #Calculo el % de  aciertos totales
  aciertos <- sum(diag(mc)) / sum(mc) * 100
  vAciertos_RF[j] = aciertos
}
cat('promedio: ',mean(vAciertos_RF),
    '%  ; min: ', min(vAciertos_RF),
    ' ; max: ', max(vAciertos_RF), '\n')

# Out:
# qvars: 33 -> promedio:  96.6119 %  ; min:  96.42857  ; max:  96.82143
# qvars: 66 -> promedio:  96.57976 %  ; min:  96.2381  ; max:  96.78571
# qvars: 132 -> promedio:  96.46548 %  ; min:  96.27381  ; max:  96.64286
```

Fig. 5: Prueba de Robustez del Modelo

3 Conclusión

La plataforma Kaggle nos permitió trabajar dandonos una gran orientación de como abordar el ánalisis, con lo cual creemos que se vuelve hasta entretenido el aprendizaje de aplicación de técnicas de ánalisis de datos, además de permitirnos competir y medirnos con otras personas. Mediante esto, hemos logrado cumplir el objetivo planteado, obteniendo un 96,564% de aciertos.

```
preds <- predict(model2, data = test)
test$Label <- as.numeric(as.character(preds$predictions))
test <- mutate(test, ImageId = row_number())
# KAGGLE:
# qvars 33 -> 0.96542
# qvars 66 -> 0.96564 --
# qvars 132 -> 0.96425
write.csv (select(test, ImageId, Label), 'submission.csv' ,fileEncoding= "UTF-8",row.names= F)
```

Fig. 6: Resultados del modelo en Kaggle

4 Anexo

4.1 Referencias

Competición Digit recognizer en Kaggle.