StreamDataMining

Carlos Manuel Sequí Sánchez

24 de abril de 2019

EJERCICIO 1 (clasificación). Se pide comparar la eficacia de un Hoeffding Tree con un clasificador Naïve Bayes, para un flujo de datos de 1.000.000 de instancias generadas con un generador RandomTreeGenerator, suponiendo una frecuencia de muestreo de 10.000 y con el método de evaluación Interleaved Test-Then-Train.

Una ejecución para Naïve Bayes:
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s generators.RandomTreeGenerator -i 1000000 -f 10000"

Una ejecución para Hoeffding Trees:
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l trees.HoeffdingTree -s generators.RandomTreeGenerator -i 1000000 -f 10000"

Para saber si hay diferencias significativas, tendríamos que generar una población de resultados y escoger una medida de eficacia para comparar. Para ello, escogeremos 30 semillas diferentes y ejecutaremos 30 veces el mismo método (en total 30 ejecuciones para Naïve Bayes y otras 30 para Hoeffding Trees). Escogeremos los resultados del porcentaje de aciertos en la clasificación, y las compararemos con un test estadístico.

A continuación una imagen del script creado para generación de la población de 30 instancias de distintas semillas con Naïve Bayes (he procedido de la misma forma para la creación de la población de Hoeffding Trees)

```
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 1) -i 1000000 -f 10000" > nb1.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 2) -i 1000000 -f 10000" > nb2.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 3) -i 1000000 -f 10000" > nb3.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 4) -i 1000000 -f 10000" > nb4.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 5) -i 1000000 -f 10000" > nb5.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 6) -i 1000000 -f 10000" > nb6.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 7) -i 1000000 -f 10000" > nb7.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 8) -i 1000000 -f 10000" > nb8.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 9) -i 1000000 -f 10000" > nb9.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 10) -i 1000000 -f 10000" > nb10.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 11) -i 1000000 -f 10000" > nb11.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 12) -i 1000000 -f 10000" > nb12.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 13) -i 1000000 -f 10000" > nb13.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 14) -i 1000000 -f 10000" > nb14.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 15) -i 1000000 -f 10000" > nb15.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 16) -i 1000000 -f 10000" > nb16.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 17) -i 1000000 -f 10000" > nb17.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 18) -i 1000000 -f 10000" > nb18.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 19) -i 1000000 -f 10000" > nb19.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 20) -i 1000000 -f 10000" > nb20.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 21) -i 1000000 -f 10000" > nb21.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 22) -i 1000000 -f 10000" > nb22.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 23) -i 1000000 -f 10000" > nb23.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 24) -i 1000000 -f 10000" > nb24.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 25) -i 1000000 -f 10000" > nb25.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 26) -i 1000000 -f 10000" > nb26.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 27) -i 1000000 -f 10000" > nb27.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 28) -i 1000000 -f 10000" > nb28.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 29) -i 1000000 -f 10000" > nb29.csv
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes -s (generators.RandomTreeGenerator -i 30) -i 1000000 -f 10000" > nb30.csv
```

Una vez generados los ficheros de datos, procedemos a crear las poblaciones en si

Leemos los datos de Naïves Bayes y creamos su población a partir de los fichers .csv

```
datosNaives = array(dim = 30)
for (i in 1:30)
{
  nombreFichero = paste(c(paste(c("./datosNaiveBayes/nb",i),collapse = ""),".csv"),collapse="")
  nuevoFichero = read.csv(nombreFichero)
  nuevoDato = nuevoFichero$classifications.correct..percent.[length(nuevoFichero$classifications.correct
```

```
  datosNaives[i] = nuevoDato
}
```

Leemos los datos de Hoeffding trees y creamos su población a partir de los fichers .csv

```
datosHoeffding = array(dim = 30)
for (i in 1:30)
{
  nombreFichero = paste(c(paste(c("./datosHoeffdingTrees/HF",i),collapse = ""),".csv"),collapse="")
  nuevoFichero = read.csv(nombreFichero)
  nuevoDato = nuevoFichero$classifications.correct..percent.[length(nuevoFichero$classifications.correc
  datosHoeffding[i] = nuevoDato
}
```

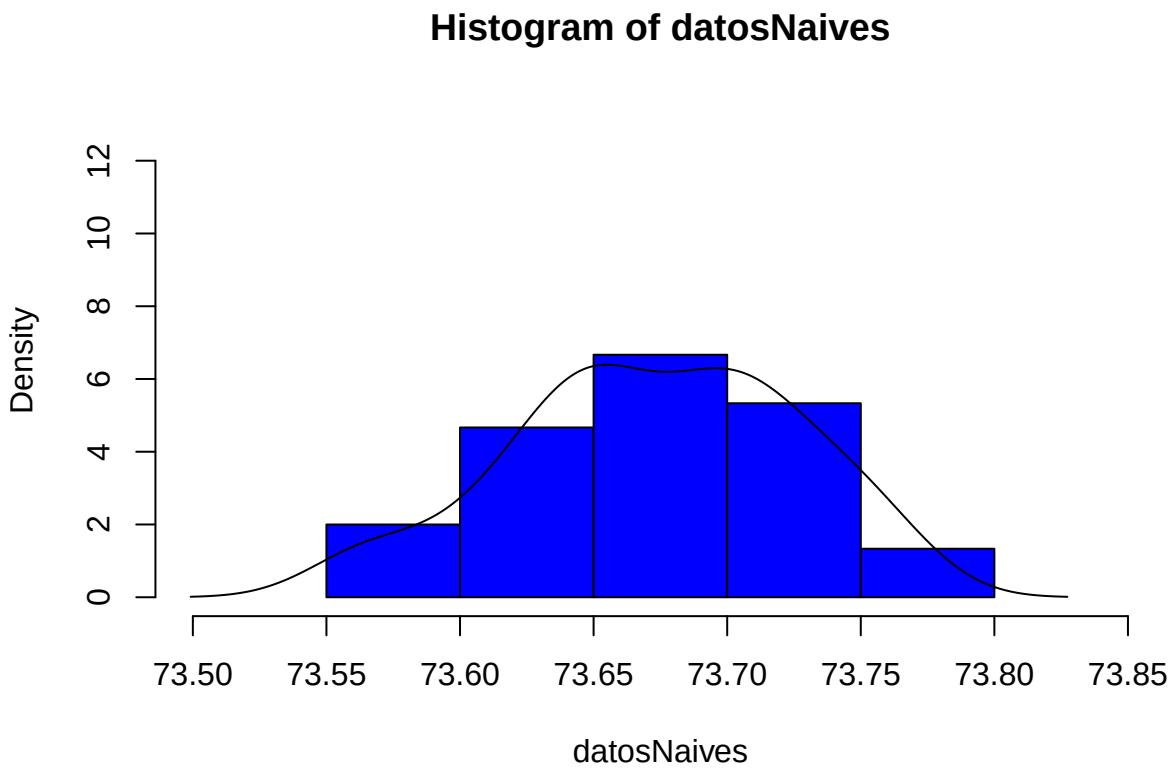
Una vez generadas las poblaciones, comprobamos la distribución de ambas para conocer que tipo de test utilizar.

Comprobamos si los datos de Naive Bayes siguen una distribución normal con el test de Saphiro Wilk y un histograma de los datos

```
shapiro.test(datosNaives)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  datosNaives
## W = 0.97381, p-value = 0.6478
```

```
hist(datosNaives, col="blue", prob=T,ylim=c(0,13), xlim=c(73.5,73.85))
lines(density(datosNaives))
```

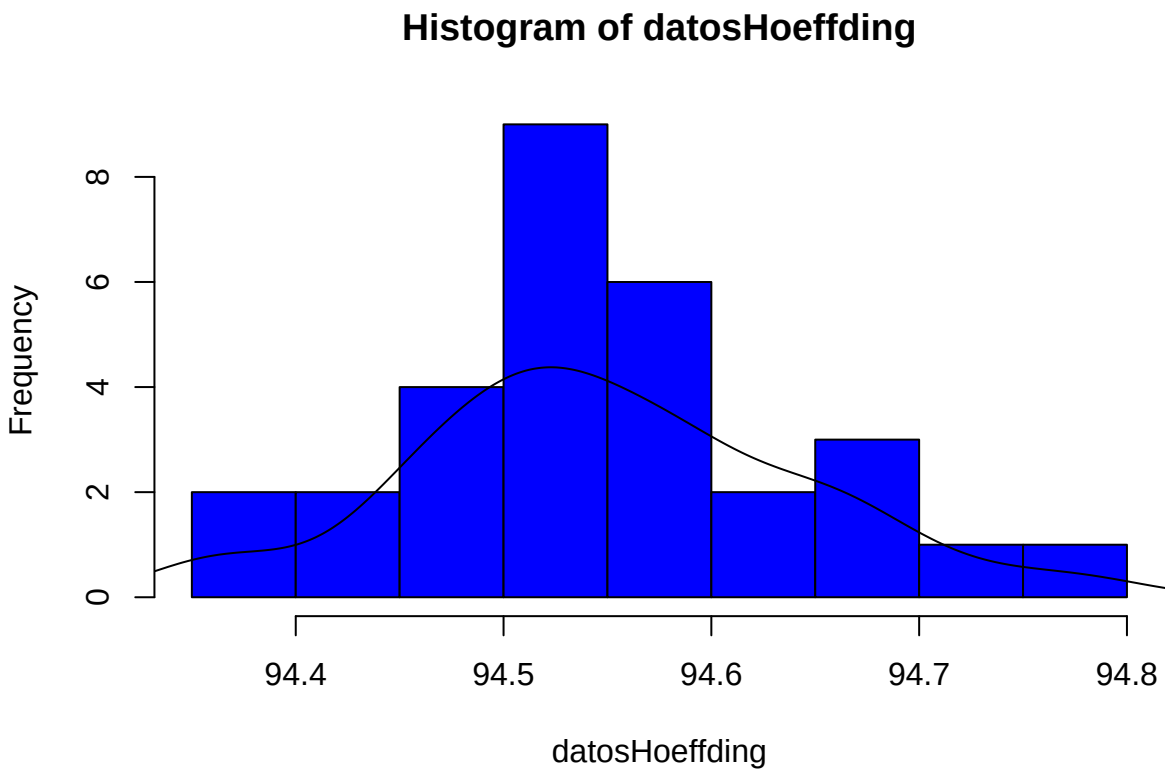


Comprobamos si los datos de Hoeffding siguen una distribución normal con el test de Saphiro Wilk y un histograma de los datos

```
shapiro.test(datosHoeffding)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  datosHoeffding
## W = 0.9824, p-value = 0.8852
```

```
hist(datosHoeffding, col="blue")
lines(density(datosHoeffding))
```



Como vemos ambos conjuntos de datos siguen una distribución normal. Debido a esto, vamos a utilizar un t-test con el fin de evidenciar si existen o no diferencias significativas entre las medias de ambos grupos.

```
t.test(datosNaives,datosHoeffding)
```

```
##
##  Welch Two Sample t-test
##
## data:  datosNaives and datosHoeffding
## t = -1055.4, df = 45.572, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -20.91581 -20.83615
## sample estimates:
## mean of x mean of y
##  73.67307  94.54905
```

Tal como observamos, al ser el p-value inferior a 0.05, podemos decir con alta certeza que las medias de las dos poblaciones de datos difiere de manera significativa. Por esta misma razón, el mejor de los algoritmos que han generado estos grupos de datos es el que mejor porcentaje de aciertos promedio ha proporcionado, es decir, el algoritmo Hoeffding Trees.

EJERCICIO 2 (Concept Drift).