SERIES TEMPORALES Y MINERÍA DE FLUJOS DE DATOS

MÁSTER EN CIENCIA DE DATOS E INGENIERÍA DE COMPUTADORES

Trabajo autónomo II: Minería de Flujo de Datos

Autora
Lidia Sánchez Mérida

Curso 2021 - 2022

Granada, Abril de 2022

Parte práctica

Entrenamiento offline (estacionario) y evaluación posterior

El objetivo de este apartado consiste en entrenar un conjunto de modelos, utilizando los algoritmos HoeffdingTree y HoeffdingAdaptiveTree, para componer dos poblaciones de métricas de calidad con las que comparar sus capacidades de predicción.

Para ello se ha definido el siguiente *script* que permite el entrenamiento y validación de un conjunto de hasta **treinta modelos por algoritmo** utilizando el software MOA desde la línea de comandos. Comienza creando la carpeta donde se almacenan las tasas de aciertos resultantes de cada uno de los clasificadores. A continuación, invocamos el programa MOA y enviamos una tarea por cada modelo a construir utilizando el comando `moa.DoTask`. Para ello especificamos el tipo de algoritmo a utilizar con el comando `LearnModel -l`. Posteriormente generamos un flujo de datos de entrenamiento compuesto por un millón de instancias con una **semilla diferente para cada clasificador** y un segundo flujo de datos del mismo tamaño con una **única semilla para validación**.

```
#!/bin/bash
mkdir -p ejercicio_2.1.1
for seed in $(seq 30)
do
   java -cp moa.jar -javaagent:sizeofag-1.0.4.jar moa.DoTask \
   "EvaluateModel -m (
       LearnModel -l trees.HoeffdingTree
       -s (generators.WaveformGenerator -i $seed) -m 1000000)
       -s (generators.WaveformGenerator -i 26) -i 1000000
   " > "./ejercicio_2.1.1/$seed.csv"
done
```

Como he comentado anteriormente, este *script* es el mismo que se ha utilizado para entrenar también hasta treinta modelos diferentes utilizando el algoritmo HoeffdingAdaptiveTree.

```
#!/bin/bash
mkdir -p ejercicio_2.1.2
for seed in $(seq 30)
do
   java -cp moa.jar -javaagent:sizeofag-1.0.4.jar moa.DoTask \
   "EvaluateModel -m (
       LearnModel -l trees.HoeffdingAdaptiveTree
       -s (generators.WaveformGenerator -i $seed) -m 1000000)
       -s (generators.WaveformGenerator -i 26) -i 1000000
   " > "./ejercicio_2.1.2/$seed.csv"
done
```

En la Figura 1 se encuentra la tabla que contiene la tasa de aciertos para cada uno de los sesenta modelos entrenados utilizando sendos algoritmos. Tal y como podemos observar, la gran mayoría de

los clasificadores generados disponen de una **precisión superior al 84%** sin fluctuaciones considerables, lo que indica que disponen de una alta estabilidad.

Semilla	HoeffdingTree	HoeffdingAdaptiveTree
1	84.4866	84.4987
2	84.5653	84.2925
3	84.5693	84.3373
4	84.4249	84.3943
5	84.3596	84.5368
6	84.4891	84.4208
7	84.6517	84.1874
8	84.5815	84.2947
9	84.5139	84.3901
10	84.4321	84.4602
11	84.5998	84.4821
12	84.4820	84.4927
13	84.5247	84.4528
14	84.5505	84.5509
15	84.6144	84.4660
16	84.5591	84.6874
17	84.7653	84.4188
18	84.6207	84.6019
19	84.6783	84.5166
20	84.5112	84.4869
21	83.4429	83.9265
22	84.5454	84.4704
23	84.6186	84.3846
24	84.4872	84.2649
25	84.5494	84.2931
26	84.4680	84.2661
27	84.3621	84.3982
28	84.7851	84.2261
29	84.1038	84.2211
30	84.5945	84.4666

Figura 1. Tabla con la tasa de aciertos de los modelos entrenados con HoeffdingTree y HoeffdingAdaptiveTree para un enfoque *offline*.

En la Figura 2 podemos observar el diagrama de cajas y bigotes relativo a las poblaciones obtenidas tras el entrenamiento y validación de treinta modelos para cada tipo de algoritmo. Según la longitud de las cajas, podemos apreciar que el modelo entrenado con **HoeffdingTree parece ser más estable** puesto que dispone de una menor variabilidad en la tasa de aciertos. Adicionalmente, la precisión media de los clasificadores asociados parece ser mayor a la media de la tasa de aciertos de los modelos entrenados con la técnica adaptativa. Por lo tanto, **gráficamente parece que el algoritmo HoeffdingTree proporciona mejores resultados** para este enfoque *offline*.

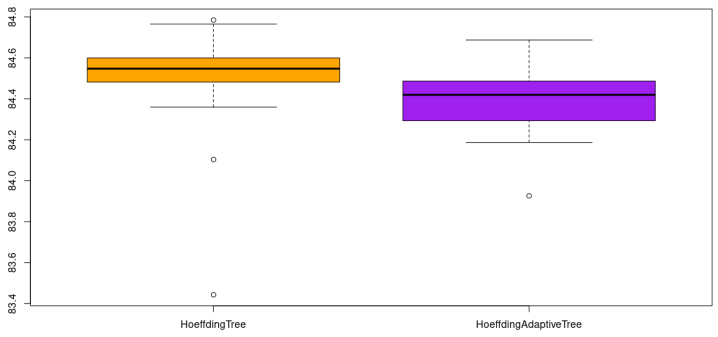

Figura 2. Diagrama de cajas y bigotes para las poblaciones *offline* de HoeffdingTree y HoeffdingAdaptiveTree.

Con el objetivo de comparar las bondades de sendos algoritmos, debemos aplicar tests estadísticos que certifiquen si existen diferencias considerables en el rendimiento de los modelos. En primer lugar, verificamos la normalidad de las poblaciones utilizando los **tests de Jarque Bera y Shapiro-Wilk**. Ambos disponen de una **hipótesis nula que apoya la teoría de que los datos siguen una distribución normal**. Estableciendo un umbral de confianza del 95%, si los p-valores resultantes son menores que 0.05 se rechaza este supuesto y por lo tanto podemos determinar que las poblaciones no son normales. En la siguiente tabla se pueden apreciar los p-valores resultantes de aplicar cada test sobre cada conjunto de datos. Tal y como podemos observar, todos los p-valores son menores o se encuentran muy próximos al umbral por lo que se rechaza la hipótesis nula y se confirma que **las poblaciones no siguen una distribución normal**.

Algoritmo	Jarque Bera	Shapiro-Wilk
HoeffdingTree	2.2e-16	4.301e-07
HoeffdingAdaptiveTree	0.02047	0.09258

Tras los resultados anteriores, a continuación aplicamos el **test no paramétrico de Wilcoxon** para muestras pareadas con el objetivo de comparar el rendimiento de ambos métodos. Su **hipótesis nula se basa en la igualdad de dos poblaciones**, por lo que utilizando el mismo umbral de confianza anterior, si el p-valor resultante es menor que 0.05 se descarta esta teoría y se confirma que sendos algoritmos disponen de comportamientos diferentes. En el caso de las técnicas HoeffdingTree y HoeffdingAdaptiveTree el **p-valor devuelto es 0.000242,** bastante inferior al umbral anterior por lo que se rechaza la hipótesis nula y se confirma que **existen diferencias significativas estadísticamente** entre ambos algoritmos. Para conocer cuál es el que proporciona clasificadores de mayor capacidad predictiva podemos calcular la media de la tasas de aciertos de los treinta modelos entrenados para cada técnica: **HoeffdingTree con una precisión media de 84.4979%** y HoeffdingAdaptiveTree con una tasa de aciertos media de 84.39622%. Por lo tanto, el algoritmo HoeffdingTree parece conseguir clasificadores de mejor calidad en base a la tasa de aciertos bajo una configuración *offline*.

Entrenamiento online

Este segundo apartado **comparte el mismo objetivo que la sección anterior,** exceptuando la naturaleza del flujo de datos con el que se pretende entrenar y validar los diferentes modelos. De nuevo se ha implementado un *script* que, en primer lugar, genera la carpeta donde se almacenarán las métricas de evaluación asociadas a los modelos construidos. Utilizando las mismas órdenes para ejecutar una tarea con el software MOA, en este ejercicio usamos el método `EvaluateInterleavedTestThenTrain` para entrenar y validar un total de treinta modelos por cada técnica. Para ello se genera un flujo de datos a partir del generador `WaveformGenerator` especificando la siguiente configuración:

- Con la primera opción `-i` establecemos una semilla diferente para cada modelo.
- La segunda opción `-i` permite especificar el número de instancias a crear en el flujo, que en el caso de este ejercicio se trata de un millón.
- Mientras que con la opción `-f` establecemos la frecuencia de muestreo.

```
#!/bin/bash
mkdir -p ejercicio_2.2.1
for seed in $(seq 30)
do
   java -cp moa.jar -javaagent:sizeofag-1.0.4.jar moa.DoTask \
   "EvaluateInterleavedTestThenTrain -l trees.HoeffdingTree
       -s (generators.WaveformGenerator -i $seed) -i 1000000 -f 10000
   " > "./ejercicio_2.2.1/$seed.csv"
done
```

Un *script* similar también ha sido diseñado para el entrenamiento de treinta modelos utilizando el algoritmo HoeffdingAdaptiveTree, tal y como se puede observar a continuación.

```
#!/bin/bash
mkdir -p ejercicio_2.2.2
for seed in $(seq 30)
do
   java -cp moa.jar -javaagent:sizeofag-1.0.4.jar moa.DoTask \
   "EvaluateInterleavedTestThenTrain -l trees.HoeffdingAdaptiveTree
       -s (generators.WaveformGenerator -i $seed) -i 1000000 -f 10000
   " > "./ejercicio_2.2.2/$seed.csv"
done
```

Como en el caso anterior, en la Figura 3 se adjunta la tabla con las tasas de aciertos asociadas a los treinta clasificadores entrenados para cada uno de los algoritmos. Para este enfoque *online* parece que se produce un **decremento de la tasa de aciertos** generalizada en todos los modelos de ambas técnicas. Por lo tanto, parece que el hecho de disponer de un flujo de datos continuo dificulta notablemente su modelado con clasificadores basados en árboles de decisión.

Semilla	HoeffdingTree	HoeffdingAdaptiveTree
1	83.8903	83.8042
2	83.8479	83.9
3	83.7456	83.7407
4	83.8392	83.7414
5	83.9761	83.8943
6	83.8801	83.8576
7	83.9843	83.8748
8	83.8343	83.8876
9	83.8963	83.7386
10	83.8406	83.7614
11	83.7880	83.7520
12	83.7851	83.7313
13	83.8152	83.8226
14	83.8623	83.8498
15	83.9778	83.8633
16	83.8462	83.6508
17	83.8860	83.8110
18	83.9650	83.8657
19	83.9033	83.8185
20	83.8202	83.8520
21	83.9	83.8143
22	83.8360	83.8220
23	83.8876	83.7875
24	83.8987	83.9733
25	84.0451	83.7961
26	83.8402	83.7144
27	83.9062	83.8406
28	83.8867	83.7784
29	83.8687	83.8968
30	83.7875	83.8282

Figura 3. Tabla con la tasa de aciertos de los modelos entrenados con HoeffdingTree y HoeffdingAdaptiveTree para un enfoque *online*.

En la Figura 4 se encuentra el diagrama de cajas y bigotes asociado a las tasas de aciertos obtenidas tras el entrenamiento de treinta modelos con el algoritmo HoeffdingTree y treinta clasificadores con HoeffdingAdaptiveTree. Se pueden apreciar dos principales diferencias con respecto a los resultados anteriores:

- Existe una **mayor variabilidad** en las precisiones obtenidas puesto que ambas cajas disponen de una mayor longitud, en comparación con el enfoque *offline* anterior. Una posible razón explicativa puede residir en que el **aumento de la complejidad de los problemas *online*** afectan directamente a la capacidad de predicción de los clasificadores entrenados con ambas técnicas. Por lo tanto, el correcto modelado de un problema cuyo flujo de datos no esté cerrado parece ser más complicado de conseguir.
- Al igual que ocurría en el caso anterior, el algoritmo HoeffdingTree es el que parece proporcionar **modelos más estables y de mejor calidad** debido a su mayor estabilidad en base a las tasas de precisión y a su rango de valores altamente superior a la técnica original.

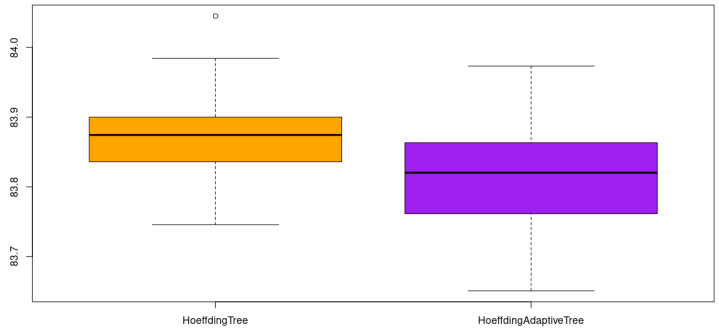

Figura 4. Diagrama de cajas y bigotes para las poblaciones *online* de HoeffdingTree y HoeffdingAdaptiveTree.

Con el fin de comparar el comportamiento de sendos algoritmos en la predicción de un problema *online*, a continuación repetimos el mismo procedimiento anterior utilizando, en primer lugar, los **tests estadísticos de Jarque Bera y Shapiro-Wilk** para comprobar la normalidad de las poblaciones. Tal y como podemos apreciar en la siguiente tabla, todos los p-valores resultantes de emplear cada método son considerablemente mayores que el umbral fijado a 0.05, lo que significa que se mantiene la hipótesis nula y se confirma que **todas las poblaciones siguen una distribución normal** al 95% de confianza.

Algoritmo	Jarque Bera	Shapiro-Wilk
HoeffdingTree	0.5001	0.3135
HoeffdingAdaptiveTree	0.9436	0.9415

A diferencia de la sección anterior, para el enfoque *online* aplicamos un **test paramétrico como es el test de Student** para muestras pareadas con el que comprobar si existen diferencias significativas entre ambos algoritmos. El **p-valor resultante es 0.001259**, muy inferior al umbral fijado en 0.05, por lo tanto se rechaza la hipótesis nula y se confirma que **existen diferencias entre los dos algoritmos**. Para conocer cuál es el que está proporcionando modelos con mayor capacidad de predicción en base a la tasa de aciertos, podemos calcular la media de las precisiones de los clasificadores entrenados con cada técnica. Así, el algoritmo **HoeffdingTree dispone de una tasa media de aciertos más elevada del 83.87468%,** mientras que la media HoeffdingAdaptiveTree es ligeramente inferior con un 83.81564%, aunque sus rendimientos son prácticamente similares.

Entrenamiento online en datos con concept drift

En el tercer ejercicio se pretende construir un modelo capaz de **detectar los cambios de concepto** que pueden surgir en un flujo de datos *online*. Nuevamente, se ha desarrollado un *script* para llevar a cabo esta tarea, cuyo primer paso consiste en generar una carpeta en la que almacenar los resultados proporcionados de los diferentes clasificadores construidos por cada algoritmo. Como en el caso anterior, aplicamos el método `EvaluateInterleavedTestThenTrain` para emplear cada nueva instancia en una primera etapa de **evaluación previo a su uso en la fase de entrenamiento**. La configuración necesaria para definir tanto el volumen como las características del flujo de datos se detallan con los siguientes parámetros:

- Con la opción `-s` construimos un flujo de datos a partir del generador `RandomRBFGeneratorDrift` estableciendo una semilla aleatoria tanto para inicializar este objeto como para producir las nuevas instancias del flujo, con las opciones `-r` y `-i`, respectivamente.
- El parámetro `-c 2` permite especificar el número de etiquetas con las que clasificar los ejemplos del flujo. Como podemos observar, en este ejercicio el problema que se trata de resolver es de clasificación binaria.
- El parámetro `-a 7` define el número de atributos que componen cada una de las muestras del flujo de datos.
- Con el argumento `-n 3` establecemos el número de centroides del modelo aplicando cambios de concepto en todos ellos mediante la segunda opción `-k 3`.
- Mediante la opción `-s 0.001` se define la velocidad con la que surgen diversos cambios de concepto.
- Finalmente, como ocurre en los ejercicios anteriores, con el parámetro `-i` fijamos el número total de ejemplos a generar, que en este caso se trata de dos millones de instancias, mientras que con el argumento `-f` establecemos la frecuencia de muestreo a cien mil.

```
#!/bin/bash
mkdir -p ejercicio_2.3.1
for seed in $(seq 30)
do
   java -cp moa.jar -javaagent:sizeofag-1.0.4.jar moa.DoTask \
   "EvaluateInterleavedTestThenTrain -l trees.HoeffdingTree
       -s (generators.RandomRBFGeneratorDrift -i $seed -r $seed
       -c 2 -a 7 -n 3 -k 3 -s 0.001) -i 2000000 -f 100000
```

```
   " > "./ejercicio_2.3.1/$seed.csv"
done
```

Existe una segunda versión del *script* anterior con una estructura prácticamente similar al anterior, a excepción del clasificador que se sustituye por un HoeffdingTree adaptativo.

```
#!/bin/bash
mkdir -p ejercicio_2.3.2
for seed in $(seq 30)
do
   java -cp moa.jar -javaagent:sizeofag-1.0.4.jar moa.DoTask \
   "EvaluateInterleavedTestThenTrain -l trees.HoeffdingAdaptiveTree
       -s (generators.RandomRBFGeneratorDrift -i $seed -r $seed
       -c 2 -a 7 -n 3 -k 3 -s 0.001) -i 2000000 -f 100000
   " > "./ejercicio_2.3.2/$seed.csv"
done
```

En la Figura 5 se recopilan las tasas de aciertos resultantes tras construir treinta clasificadores de cada técnica con la configuración explicada anteriormente. Tal y como podemos observar, existe una **mayor variabilidad** en los valores de sendos métodos, aunque especialmente notoria en el caso del *HoeffdingTreee*, ya que esta técnica no incorpora **ninguna metodología para detectar los cambios de concepto.** Sin embargo, la versión adaptativa utiliza una ventana deslizante con la que es capaz de controlar la precisión de cada rama del árbol y reemplazar aquellas con una menor precisión.

Semilla	HoeffdingTree	HoeffdingAdaptiveTree
1	77.45465	96.24925
2	65.65775	87.76635
3	74.93125	96.0234
4	59.5718	85.98955
5	68.8432	94.35425
6	76.17895	92.0137
7	96.9101	97.3471
8	80.4608	97,432
9	70.9067	93.33335
10	97.5558	97.7868
11	84.6607	92.18935
12	83.95655	92.64715
13	76.8663	95.6582
14	68.99015	86.54875
15	75.34355	89.69125
16	77.00125	96.1635
17	67.8415	92.21095
18	65.7344	85.5335
19	71.68625	97.1743
20	69,609	90.3818
21	71.89805	94.39065
22	77.99515	96.5747
23	73.4734	96.3097
24	61.5014	91.46245
25	73.0467	95.9832
26	72.06165	90.99615
27	83.37765	94.9591
28	61,572	88.70825
29	78.38975	90.8219
30	98.1665	98.54465

Figura 5. Tabla con la tasa de aciertos de los modelos entrenados con HoeffdingTree y HoeffdingAdaptiveTree para un enfoque *online* con *concept drift*.

En las dos siguientes imágenes (Figura 6 y Figura 7) se representan las evoluciones de dos modelos entrenados con las diferentes técnicas contempladas hasta el momento, HoeffdingTree y su versión adaptativa, respectivamente. En el primer caso podemos observar cómo la precisión refleja una **función decreciente**, en la que presumiblemente disminuye la precisión conforme aparecen los cambios de concepto en el flujo de datos, mientras que en la segunda gráfica la **precisión se mantiene constante** y alrededor de valores considerablemente más altos en el eje Y. Ambos comportamientos demuestran la importancia de utilizar un algoritmo adecuado a partir del estudio de las características tanto relativas al problema como al flujo de datos a modelar.

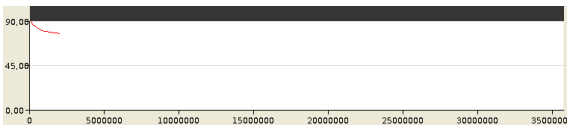

Figura 6. Evolución de la curva de aciertos del algoritmo HoeffdingTree.

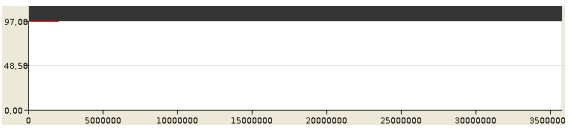

Figura 7. Evolución de la curva de aciertos del algoritmo HoeffdingAdaptiveTree.

Después del estudio gráfico anterior en el que parece que el algoritmo HoeffdingAdaptiveTree es capaz de generar modelos con mayor capacidad de generalización, a continuación procedemos a aplicar los **tests estadísticos de Jarque Bera y Shapiro-Wilk** para comprobar la normalidad de ambas poblaciones. En función de los p-valores podemos determinar que al ser mayores o considerablemente cercanos al umbral definido en 0.05 la hipótesis nula no se rechaza y por ende se confirma que las dos **poblaciones siguen una distribución normal** con un 95% de confianza estadística.

Algoritmo	Jarque Bera	Shapiro-Wilk
HoeffdingTree	0.1647	0.03698
HoeffdingAdaptiveTree	0.3386	0.06625

A continuación empleamos un **test paramétrico como el test de Student** para muestras pareadas con el objetivo de verificar si ambos algoritmos se comportan de manera diferente, tal y como parece ocurrir a partir del análisis anterior. El **p-valor devuelto es 3.463e-11** por lo que al ser sumamente menor al umbral establecido en 0.05 podemos confirmar que **existen diferencias estadísticamente significativas** entre ambas técnicas sobre un enfoque *online* con cambios de concepto. Nuevamente, calculando la media de las tasas de aciertos para los modelos entrenados con cada algoritmo podemos observar que HoeffdingTree dispone de un 75.3881% de precisión, mientras que **HoeffdingAdaptiveTree alcanza una cifra muy superior con un 93.17484% de aciertos.** Por lo tanto, se acepta la teoría previa que confirma el altísimo rendimiento que presenta un algoritmo adaptativo sobre un flujo de datos *online* con modificaciones sustanciales en el tiempo, mientras que la versión original no es capaz ni de identificar ni de adaptarse a la nueva información entrante, lo que produce que los modelos resultantes no dispongan de una buena capacidad de predicción.

Entrenamiento online en datos con concept drift, incluyendo mecanismos para olvidar instancias pasadas

La cuarta sección trata de replicar el procedimiento anterior intercambiando el método de entrenamiento y validación de los modelos por **Prequential**, con el que también se aplica un enfoque *test-then-train* aunque considerando únicamente los **datos más recientes** a partir de una **ventana deslizante** que almacena un total de mil instancias. Por lo tanto el *script* es prácticamente similar al que aparecía en primer lugar dentro de la sección anterior, a excepción del criterio mencionado. En este caso establecemos el método `EvaluatePrequential` con la opción `-e` para establecer una ventana deslizante haciendo referencia a la clase `WindowClassificationPerformanceEvaluator` con un tamaño de mil datos mediante el argumento `-w 1000`.

```
#!/bin/bash
mkdir -p ejercicio_2.4.1
for seed in $(seq 30)
do
   java -cp moa.jar -javaagent:sizeofag-1.0.4.jar moa.DoTask \
   "EvaluatePrequential -l trees.HoeffdingTree
       -e (WindowClassificationPerformanceEvaluator -w 10000)
       -s (generators.RandomRBFGeneratorDrift -i $seed -r $seed
       -c 2 -a 7 -n 3 -k 3 -s 0.001) -i 2000000 -f 100000
   " > "./ejercicio_2.4.1/$seed.csv"
done
```

De forma similar se ha diseñado un segundo *script* para ejecutar la misma configuración explicada en esta sección para un algoritmo HoeffdingTree adaptativo.

```
#!/bin/bash
mkdir -p ejercicio_2.4.2
for seed in $(seq 30)
do
   java -cp moa.jar -javaagent:sizeofag-1.0.4.jar moa.DoTask \
   "EvaluatePrequential -l trees.HoeffdingAdaptiveTree
       -e (WindowClassificationPerformanceEvaluator -w 10000)
       -s (generators.RandomRBFGeneratorDrift -i $seed -r $seed
       -c 2 -a 7 -n 3 -k 3 -s 0.001) -i 2000000 -f 100000
   " > "./ejercicio_2.4.2/$seed.csv"
done
```

A continuación, en la Figura 8 se encuentran representadas las precisiones de los treinta modelos construidos con cada uno de los algoritmos de árboles de decisión estudiados hasta el momento. El comportamiento demostrado en base a esta métrica de calidad es **prácticamente similar al del ejercicio anterior**. Mientras que los clasificadores generados con *HoeffdingTree* se caracterizan por una notable variabilidad en las tasas de aciertos, alcanzando como máximo un 75% de precisión, en cambio los modelos entrenados con la versión adaptativa son bastante más estables y disponen de tasas de aciertos por encima del 90%.

Semilla	HoeffdingTree	HoeffdingAdaptiveTree
1	71.28	97.8
2	61.3	86.52
3	74.64	97.58
4	57.97	85.99
5	65.58	94.32
6	73.42	92.22
7	96.85	97.08
8	76.12	98.32
9	69.25	94.04
10	97.57	98.55
11	84.62	91.81
12	83.77	94.96
13	74.9	95.53
14	69.59	84.48
15	73.81	90.71
16	72.48	95.99
17	67	90.83
18	67.19	88.77
19	72.42	96.9
20	72.04	92.96
21	70.64	96.07
22	74.61	96.09
23	67.16	96.4
24	56.15	87.91
25	64.58	96.02
26	70.27	91.43
27	82.34	93.94
28	57.94	86.15
29	79.12	91.78
30	98.17	98.96

Figura 8. Tabla con la tasa de aciertos de los modelos entrenados con HoeffdingTree y HoeffdingAdaptiveTree para un enfoque *online* con *concept drift* utilizando *Prequential* como criterio de entrenamiento y evaluación.

Si observamos las gráficas dibujadas en la Figura 9 y Figura 10 podemos apreciar la evolución de la curva de entrenamiento de dos modelos construidos con HoeffdingTree y su variante, respectivamente. La **tendencia representada es notablemente parecida** a la analizada en el ejercicio anterior, por lo que se confirma el buen rendimiento del algoritmo adaptativo frente a la inestabilidad y baja precisión por parte del clasificador generado mediante la técnica original. No obstante, comparando este par de gráficas con las relativas al enfoque clásico de *test-then-train* sí que se aprecia un **mayor desequilibrio** en la capacidad de generalización de estos modelos que utilizan una ventana deslizante, debido a la mayor estabilidad que sufren sus respectivas líneas. Este hecho pone de manifiesto la **diferencia existente entre ambos criterios de evaluación**, siendo el más beneficioso el Test-Then-Train clásico frente al Prequential. Una posible razón explicativa de este fenómeno reside en que, considerando un flujo de datos de dos millones de instancias, establecer el **tamaño de la ventana** deslizante únicamente en mil instancias puede ser perjudicial en caso de que no existan suficientes datos como para que los modelos converjan rápidamente hacia una solución.

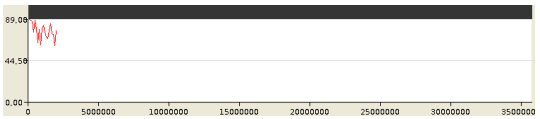

Figura 9. Evolución de la curva de aciertos del algoritmo HoeffdingTree con *prequential*.

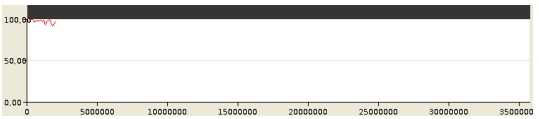

Figura 10. Evolución de la curva de aciertos del algoritmo HoeffdingAdaptiveTree con *prequential*.

En la siguiente tabla se localizan los p-valores obtenidos tras aplicar los **tests estadísticos de Jarque Bera y Shapiro-Wilk** para verificar si las dos poblaciones generadas a partir de la tasas de aciertos de los diferentes clasificadores siguen una distribución normal. Como podemos observar,

los valores resultantes son muy similares a los de la sección anterior por lo que, nuevamente, podemos determinar que **los datos son normales**.

Algoritmo	Jarque Bera	Shapiro-Wilk
HoeffdingTree	0.1605	0.02255
HoeffdingAdaptiveTree	0.284	0.04334

Finalmente hacemos uso del **test paramétrico de Student** para muestras pareadas con el que comparar si existen diferencias significativas entre las técnicas HoeffdingTree y su versión adaptativa. Como el p-valor resultante es 1.092e-09, notablemente menor que el umbral fijado a 0.05, se confirma que los **dos algoritmos presentan comportamientos distintos**. Tal y como sucedía en el ejercicio anterior, la técnica **HoeffdingAdaptiveTree dispone de una precisión media del 93.337%** mientras que los modelos generados con el algoritmo original se encuentran alrededor del 73.426%. Como consecuencia, tras utilizar diferentes enfoques para construir varios clasificadores tanto en el ejercicio anterior como en este presente, confirmamos la superioridad que presenta el algoritmo adaptativo para el modelado de flujos de datos *online* caracterizados por la aparición de cambios de concepto, independientemente del criterio de entrenamiento y evaluación.

Entrenamiento online en datos con concept drift, incluyendo reinicializar modelos tras la detección de cambios de concepto

Este último apartado trata de construir clasificadores con la **misma configuración del tercer apartado**, es decir, a partir de un flujo de datos *online* con la aparición de varios cambios de concepto, aunque incluyendo técnicas adicionales para **reemplazar aquellos clasificadores** que proporcionen menores tasas de aciertos debido a modificaciones considerables en la información recibida. Al igual que en los apartados previos, se han desarrollado dos *scripts* semejantes a los del ejercicio 2.3. para utilizar tanto el algoritmo HoeffdingTree como su variante adaptativa. A continuación podemos observar el primero de ellos, cuya principal diferencia consiste en añadir la clase `drift.SingleClassifierDrift`, que permite la monitorización del número de errores que comete el modelo generado con HoeffdingTree para detectar si ha surgido un cambio de concepto en base a un aumento considerable de la tasa de fallos. Este criterio se encuentra asociado al detector **DDM** seleccionado mediante el argumento `-d DDM`.

```
#!/bin/bash
mkdir -p ejercicio_2.5.1
for seed in $(seq 30)
do
   java -cp moa.jar -javaagent:sizeofag-1.0.4.jar moa.DoTask \
   "EvaluateInterleavedTestThenTrain -l drift.SingleClassifierDrift -d DDM
       -l trees.HoeffdingTree
       -s (generators.RandomRBFGeneratorDrift -i $seed -r $seed
       -c 2 -a 7 -n 3 -k 3 -s 0.001) -i 2000000 -f 100000
   " > "./ejercicio_2.5.1/$seed.csv"
done
```

Tal y como se ha comentado anteriormente, el segundo *script* comparte la estructura completa del primero visualizado con la particularidad de aplicar el algoritmo adaptativo HoeffdingAdaptiveTree.

```
#!/bin/bash
mkdir -p ejercicio_2.5.2
for seed in $(seq 30)
do
   java -cp moa.jar -javaagent:sizeofag-1.0.4.jar moa.DoTask \
   "EvaluateInterleavedTestThenTrain -l drift.SingleClassifierDrift -d DDM
       -l trees.HoeffdingAdaptiveTree
       -s (generators.RandomRBFGeneratorDrift -i $seed -r $seed
       -c 2 -a 7 -n 3 -k 3 -s 0.001) -i 2000000 -f 100000
   " > "./ejercicio_2.5.2/$seed.csv"
done
```

Las tasas de aciertos resultantes de los treinta modelos generados para cada algoritmo se pueden visualizar en la Figura 11. De nuevo, apreciamos una tendencia de **variabilidad** similar en los valores asociados a los clasificadores entrenados con la técnica HoeffdingTree, al igual que ocurría en las dos secciones anteriores, mientras que por otro lado parece haber una **mayor estabilidad y robustez** en los modelos construidos a partir del algoritmo de árboles de decisión adaptativo por la alta semejanza que presentan sus respectivas precisiones.

Semilla	HoeffdingTree	HoeffdingAdaptiveTree
1	77.45465	96.24925
2	65.65775	87.76635
3	74.93125	96.0234
4	59.5718	85.98955
5	68.8432	94.35425
6	76.17895	92.0137
7	96.9101	97.3471
8	80.4608	97432
9	70.9067	93.33335
10	97.5558	97.7868
11	84.6607	92.18935
12	83.95655	92.64715
13	76.8663	95.6582
14	68.99015	86.54875
15	75.34355	89.69125
16	77.00125	96.1635
17	67.8415	92.21095
18	65.7344	85.5335
19	71.68625	97.1743
20	69609	90.3818
21	71.89805	94.39065
22	77.99515	96.5747
23	73.4734	96.3097
24	61.5014	91.46245
25	73.0467	95.9832
26	72.06165	90.99615
27	83.37765	94.9591
28	61572	88.70825
29	78.38975	90.8219
30	98.1665	98.54465

Figura 11. Tabla con la tasa de aciertos de los modelos entrenados con HoeffdingTree y HoeffdingAdaptiveTree para un enfoque *online* con *concept drift* y mecanismos para reiniciar los modelos.

Procedemos a identificar si existen diferencias estadísticamente significativas entre los modelos entrenados por ambas técnicas aplicando, en primer lugar, los **tests estadísticos de Jarque Bera y Shapiro-Wilk** para evaluar si sus poblaciones siguen una distribución normal. Como podemos visualizar en la siguiente tabla, de nuevo los p-valores obtenidos son mayores o considerablemente

cercanos al umbral mínimo de 0.05 por lo que se confirma que sendas **poblaciones siguen una distribución normal** al 95% de confianza.

Algoritmo	Jarque Bera	Shapiro-Wilk
HoeffdingTree	0.1647	0.03698
HoeffdingAdaptiveTree	0.3386	0.06625

Tras confirmar la normalidad de los datos empleamos el **test paramétrico de Student** para muestras pareadas con el objetivo de determinar si los dos algoritmos son semejantes. Como el p-valor devuelto es 3.463e-11, sumamente inferior al umbral fijado en 0.05 podemos afirmar que **las dos técnicas son diferentes** para este último enfoque. De nuevo, tal y como era de esperar, el **algoritmo adaptativo muestra un 93.17484% de precisión**, mientras que el método original se encuentra cerca del 75.3881% de aciertos, un valor sumamente inferior al proporcionado por la técnica HoeffdingAdaptiveTree.

Para terminar este trabajo en la siguiente tabla se recopilan las **tasas de aciertos medias** de los treinta modelos generados para cada algoritmo en cada una de las tres configuraciones practicadas para la detección de cambio de concepto. En la técnica **HoeffdingTree**, parece que el uso de la **ventana deslizante no ha sido beneficioso** puesto que se trata de la menor precisión conseguida con este método. Tal y como se detalló anteriormente, presumiblemente se pueda intentar corregir este comportamiento decreciente aumentando el tamaño de la ventana de modo que exista un mayor número de datos que facilite la convergencia de los modelos. Por otro lado, **no existen prácticamente diferencias** significativas entre las distintas técnicas aplicadas sobre los modelos construidos con **HoeffdingAdaptiveTree**. Como consecuencia podemos intuir que el buen comportamiento de estos clasificadores en base a las tasas de aciertos se encuentra estrechamente ligado a la capacidad del algoritmo para detectar y modificar el modelado del flujo de datos cuando surge un cambio de concepto.

Algoritmo	Concept drift	Concept drift + ventana	Concept drift + Reemplazo
HoeffdingTree	75.3881%	73.426%	75.3881%
HoeffdingAdaptiveTree	93.17484%	93.337%	93.17484%