UNIVERSIDAD DE GRANADA

MÁSTER DE CIENCIA DE DATOS E INGENIERÍA DE COMPUTADORES

CURSO ACADÉMICO 2019-2020

SERIES TEMPORALES Y MINERÍA DE FLUJO DE DATOS

Trabajo autónomo II: Minería de Flujos Temporales.

Estudio, desarrollo y comparación de modelos de clasificación de Flujos Temporales.

Nicolás Cubero

10 de Mayo de 2020

Índice

Índice de figuras

1. Fundamentos teóricos de Minería de Flujos temporales

En este apartado, se tratarán todos los fundamentos teóricos sobre **Minería de Flujos Temporales** y sobre los métodos, algoritmos y clasificadores necesarios para desarrollar y entender este proyecto.

1.1. Resolución del cuestionario del anexo

Primeramente, se tratará la resolución del cuestionario de preguntas tipo test del anexo:

1. **El aprendizaje incremental es útil cuando ...**
   se quiere ganar eficiencia.

2. **La mineria de flujo de datos se considera cuando...**
   el problema genera datos continuamente.

3. **La cota de Hoeffding sirve para saber...**
   cuando hay suficientes datos para una estimación fiable.

4. **¿Que caracterısticas de clústeres mantiene el algoritmo BIRCH?**
   Suma lineal, suma cuadratica y numero de objetos.

5. **¿El algoritmo Stream maneja concept drift?**
   No.

6. **¿Que es concept drift?**
   Cambios en la dinámica del problema.

7. **¿Como gestiona CVFDT el concept drift?**
   Mantiene árboles alternativos.

8. **¿Por qué es útil el ensemble learning en concept drift?**
   Porque aprovecha la diversidad que se genera en los cambios.

9. **¿Cual es mas eficiente entre DDM y ADWIN?**
   DDM es mas eficiente.

10. **¿Por que es controvertida la clasificacion en flujo de datos?**
    Porque se requiere al oráculo por siempre.

11. **¿Como gestiona CluStream el concept drift?**
    Mantiene información sobre el tiempo.

12. **¿Por qué es complejo generar reglas de asociacion en flujo de datos**
    Porque las reglas se evalúan sobre el histórico de datos.

1.2. Clasificación de flujos temporales

Un **flujo de datos** es una secuencia **ordenada** y **continua** de información que es recibida en una **tasa muy elevada** y con una **distribución no estacionaria y cambiante**.

Al ser de naturaleza continua y tener una tasa de llegada elevada, imposibilita su almacenamiento y obliga a que su explotación se realice en tiempo real y que después sean desechados, es decir, la explotación de los flujos temporales se realizan bajo unas capacidades de cómputo y almacenamiento reducidos.

La **Minería de flujo de datos** constituye un campo de **Minería de datos** que implementa y/o redefine métodos y algoritmos para la extración de conocimiento implícito y de interés de este flujo de datos de forma dinámica conforme los datos se van recibiendo.

La **clasificación de flujos de datos** constituye uno de los problemas tratados en **Minería de flujo de datos** que, partiendo de un flujo continuo de datos etiquetado, trata de elaborar modelos de clasificación capaces de ajustarse de forma continua a los datos que se reciben y tratar de predecir las etiquetas de datos futuros.

Para la clasificación de flujos de datos, se harán uso de los siguientes modelos:

- **Hoeffding Tree** o **Very Fast Decision Tree** (VFDT): Modelo incremental basado en árboles de clasificación que, construye cada nodo de decisión estableciendo en los mismos un criterio de separación que particiona el conjunto de datos de forma que se maximice la ganancia de información (modelo ID3) o el ratio de la ganacia (modelo C4.5).

  Dado que no almacena todos los datos que recibe, calcula esta ganancia de forma local con los datos recibidos en cada instante para cada

atributo y teniendo en cuenta estadísticos calculados con datos anteriores, y hace uso de **la desigualdad de de *Hoeffding*** para tratar de determinar si el particionamiento en base a un valor de atributo en un nodo de decisión, proporciona una ganancia suficiente para predecir instancias futuras.

Este clasificador no implementa ningún mecanismo para desechar ajustes anteriores a datos previos a un cambio de concepto (*concept driff*), por lo que su rendimiento puede no ser óptimo para flujos de datos con *concept driff*.

Por último, el clasificador constituye un **modelo de clasificación incremental** más que un modelo diseñado para trabajar propiamente con flujo de datos, es decir, se halla implementado para trabajar con datos estáticos que se reciben de forma progresiva pero no con la continuidad propia de un flujo de datos y requiere además, almacenar parte de los datos que recibe. Por todo ello, **constituye un clasificador con ciertas limitaciones para trabajar con flujos de datos**.

- **Hoeffding Adaptive Tree**: constituye un modelo de árbol de clasificación basado en *Hoeffding Tree*, que, a diferencia de este, implementa un método para adaptarse a los cambios de concepto y desechar el ajuste a los datos previos al cambio de concepto.

  Para ello, hace uso de una ventana **ADWIN** (*ADaptive sliding WINdow*), para detectar las ramas del árbol que ya no ajustan a los datos tras un cambio de concepto y eliminarlas, al tiempo que la definición de nuevos nodos de decisión en base a los datos posteriores que se reciben constituyen las nuevas ramas que ajustan a los datos posteriores a cada cambio de concepto.

  De este modo, este clasificador permite ajustarse de forma dinámica a los datos posteriores a cada cambio de concepto y "olvidar" el ajuste realizado a los datos anteriores.

- **Single Classifier Drift**: Método implementado en el software MOA que hace uso de un detector de *concept drift* para determinar cuando se produce un cambio de concepto en el flujo de datos, y comenzar el entrenamiento de un nuevo clasificador con los datos posteriores al cambio de concepto.

Por su parte, para ajustar y evaluar estos modelos con los datos que se

van recibiendo (y que en este proyecto se generarán de forma automática haciendo uso de generadores), se utilizan los siguientes mecanismos:

- **Interleaved Test-Then-Train**: Entrena un modelo con los datos que se van recibiendo de un flujo de datos pero con cada instancia que se recibe del flujo, evalúa primeramente su error y calcula el rendimiento sobre todas las instancias recibidas hasta el momento, acto seguido, se usa la nueva instancia para alimentar y entrenar el modelo.

- **Evaluación prequencial**: Al igual que el método *Interleaved Test-Then-Train*, con cada instancia que se recibe del flujo, se evalúa el rendimiento del modelo y después se usa para alimentar el, pero a diferencia de *Interleaved Test-Then-Train*, el cálculo del rendimiento no se realiza sobre todas las instancias, sino sobre las instancias más recientes que se incluyen en una ventana deslizante, o bien, haciendo uso de un factor de decaimiento que da relevancia a las instancias más recientes.

1.3. Detección del *Concept Drift*

Concept Drift o **cambio de concepto** es un fenómeno en los flujos de datos consistente en una alteración de la ley subyacente de los datos tras determinados periodos de tiempo como consecuencia de ruido, cambios estacionarios, influencias del entorno, etc que provoca que los modelos entrenados ya no sean consistente con la nueva ley subyacente en el flujo.

La aparición de cambios de concepto en los datos lleva al diseño de clasificadores y/o métodos capaces de detectar estos fenómenos y ajustar los modelos a los nuevos datos dando menor relevancia y/o desechando los ajustes realizados con los anteriores datos.

Para la detección del cambio de concepto se encuentran disponibles los siguientes métodos:

- **Aprendizaje online**: Se refiere a mecanismos implementados en clasificadores para detectar los cambios de concepto, ajustar y/o dar mayor relevancia al ajuste a los datos posteriores al cambio de concepto y desechar los ajustes anteriores.

  Uno de los clasificadores más destacados es *Concept-adapting Very Fast Decision tree* (CVFDT) que genera y mantiene ramas alternativas para los datos posteriores a un cambio de concepto, evalúa la validez de las

ramas originales reemplazando las ramas que ajustan peor a los datos por las ramas alternativas.

- **Métodos de ventana deslizante**: Estos métodos hacen uso de una ventana deslizante donde se incluyen las instancias más recientes. Sólo las instancias incluídas en la ventana deslizante son empleadas en el aprendizaje de los clasificadores, las instancias anteriores son excluídas de esta ventana y, por lo tanto, no se entrena con ellos.

  Este método puede incluir además un factor de ponderación por el cual se otorgue mayor relevancia a las instancias más recientes que a las instancias más antiguas.

- **Métodos de *Ensemble***: Estos métodos construyen un *ensemble* de clasificadores elementales de tamaño es fijo. Cada vez que llega un número suficiente de instancias, se construye un nuevo clasificador que reemplaza a otro clasificador anterior que ofrece la peor precisión del *ensemble*.

  La clasificación de una nueva instancia se realiza mediante votación a partir de las clasificaciones de cada clasificador individual. Se evalúa periodicamente la precisión de cada clasificador individual de modo que el clasificador con peor rendimiento es reemplazado por un nuevo clasificador que se genere.

  Dentro de esta propuesta destacan ***Streaming Ensemble Algorithm*** (SEA) en el que la clasificación se realiza por voto mayoritario de los clasificadores individuales, ***Accuracy Weighted Ensemble*** **(AWE)** donde la clasificación se realiza por voto ponderado de los clasificadores individuales y ***Dinamic Weighted Majority*** (DWM) que pondera los votos según las clasificaciones incorrectas de cada clasificador.

- **Métodos de detección de *Concept Drift***: Constituye una serie de algoritmos que tratan de determinar cuando se produce *Concept Drift* en el flujo de datos.

  De entre estos algoritmos destacan los siguientes:

  - ***Drift detection method*** (DDM): Detecta un cambio de concepto cuando se produce una disminución en la precisión del clasificador entrenado respecto de los datos recibidos, o bien cuando se produce un cambio en la distribución de las clases.

Para ello, para cada instancia i determina la probabilidad de cometer un error de clasificación p_i y determina la desviación estándar s_i esperada de esta probabilidad de error:

$$s_i = \sqrt{p_i * (1 - p_i)/i} \tag{1}$$

Si el error cometido en una determinada instancia sobrepasa un umbral fijado tomando como referencia el ratio de error mínimo cometido p_{min} con la desviación mínima de este ratio de error mínomo (s_{min}), se asume que se produce *concept drift*.

Se fijan dos umbrales o niveles:

- Nivel de advertencia ($p_{min} + 2 * s_{min}$): Si se da que $p_i + s_i \geq p_{min} + 2 * s_{min}$, se alcanza el nivel de advertencia y se comienzan a almacenar las instancias del flujo por si constituyeran datos posteriores a un cambio de concepto.
- Nivel *Drift*: ($p_{min} + 3 * s_{min}$): Si se da que $p_i + s_i \geq p_{min} + 3 * s_{min}$, entonces se considera que se ha producido un cambio de concepto .

- ***Adaptive sliding Window*** (ADWIN): Toma como referencia una ventana W donde incluye las instancias más recientes recibidas del flujo y realiza diversas particiones de esta ventana en dos subconjuntos W_0 y W_1. Si en alguna de estas particiones la media de la precisión de ambos subconjuntos difiere más de un umbral establecido, se considera que se ha producido *Concept Drift*.

  Esta mecanismo es el usado por el clasificador ***Hoeffding Adaptive Tree***, para analizar cuales ramas de su árbol ya no ajustan a los datos tras el cambio de concepto y eliminarlas.

- ***Histogram-based straightforward prediction*** (HSP): Método que se basa en la detección del cambio en la distribución de las clases predichas para determinar cuando se produce un cambio de concepto. Modela un histograma con la distribución de las clases predichas para monitorizar el límite de decisión, de forma que, ante un cambio en la distribución de las clases predichas, el límite de decisión sufre variaciones gracias a las cuales se detecta este cambio de distribución y, por consiguiente, el *concept drift*.

  La definición de este algoritmo resulta con una menor complejidad

en tiempo y en espacio que los métodos de detección de *Concept Drift* anteriores, lo que resulta en un método más eficiente.

2. Desarrollo práctico

A continuación, se desarrollarán una serie de ejercicios donde se entrenarán y analizarán los rendimientos de diferentes modelos de Minería de datos sobre diferentes flujos de datos generados con generadores automáticos.

Los experimentos se ejecutarán con ayuda del software ***MOA*** (*Massive Online Analysis*).

2.1. Entrenamiento offline (estacionario) y evaluación posterior

2.1.1. Entrenar un clasificador HoeffdingTree offline (estacionario, aprender modelo únicamente), sobre un total de 1.000.000 de instancias procedentes de un flujo obtenido por el generador WaveFormGenerator con semilla aleatoria igual a 2.Evaluar posteriormente (sólo evaluación) con 1.000.000 de instancias generadas por el mismo tipo de generador, con semilla aleatoria igual a 4. Repita el proceso varias veces con la misma semilla en evaluación y diferentes semillas en entrenamiento, para crear una población de resultados. Anotar como resultadoslos valores de porcentajes de aciertos en la clasificación y estadístico Kappa

Para la ejecución de este experimento desde la línea de comandos de *MOA*, procederíamos a introducir el siguiente comando:

```
EvaluateInterleavedTestThenTrain -l (LearnModel -l trees.HoeffdingTree
                                      -s (generators.WaveformGenerator -i 2)
                                      -m 1000000)
-s (generators.WaveformGenerator -i 4)
-i 1000000
```

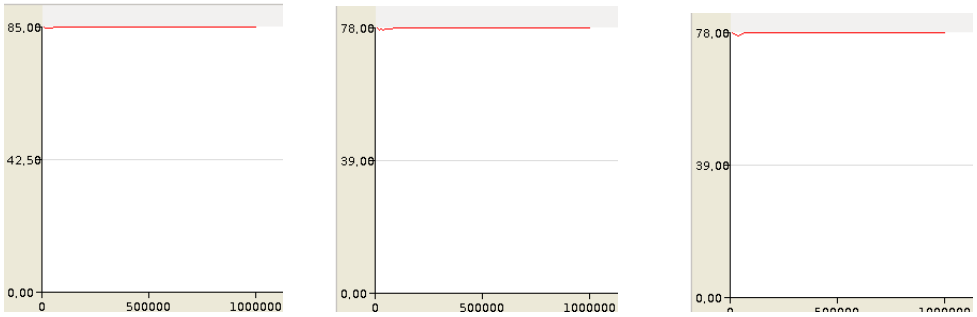
Con este comando, ejecutaremos un método de evaluación y entrenamiento *EvaluateInterleavedTestThenTrain* que trabajaría con 1.000.000 de

instancias generadas (parámetro `-i`) para evaluar el clasificador preentrenado.

Estas instancias son generadas mediante el generador *WaveformGenerator* (especificado mediante el parámetro `-s`). Este generador generará instancias aleatorias que se usarán para evaluar el modelo preentrenado usando un valor de 4 como semilla (parámetro `-i` del generador).

Por último, respecto al modelo evaluado (parámetro `-l`), se basa en un clasificador *HoeffdingTree* estacionario (parámetro `-l` de la utilidad *LearnModel*) que es entrenado con otro flujo de instancias generadas también con un generador *Waveformgenerator* (parámetro `-s` de la utilidad *LearnModel*), esta vez, con un valor de semilla 2 (parámetro `-i` de este generador).

Ejecutamos esta experimentación obteniendo la siguiente evolución de las métricas:



(a) Accuracy (b) Estadístico Kappa (c) Kappa Temporal

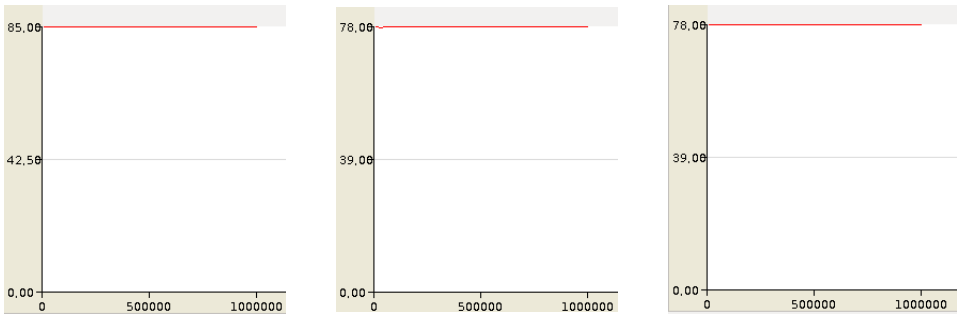
Figura 1: Representación gráfica de la evolución de las métricas de Accuracy, Kappa y Kappa Temp del clasificador HoeffdingTree estacionario entrenado con un generador WaveformGenerator considerando una semilla aleatoria de 2.

Se observa que el rendimiento de este clasificador se mantiene constante y con un rendimiento óptimo. Las métricas de este clasificador en el último instante son las siguientes:

semilla	Medida	Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
2	Última instancia	84.7239	77.08758286928095	77.0957904269547
2	Media	84.7	77.05	77.05

Tabla 1: Métricas de rendimiento del clasificador HoeffdingTree estacionario entrenado con un generador WafeformGenerator con seed 2

Repetimos esta experimentación estableciendo como semilla el valor 11 para el generador usado para el entrenamiento del clasificador:



(a) Accuracy (b) Estadístico Kappa (c) Kappa Temporal

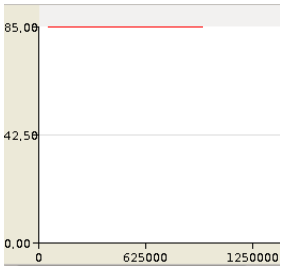
Figura 2: Representación gráfica de la evolución de las métricas de Accuracy, Kappa y Kappa Temp del clasificador HoeffdingTree estacionario entrenado con un generador WaveformGenerator considerando una semilla aleatoria de 11.

Se observa un rendimiento similar al de la anterior experimentación. Las métricas con esta semilla resultan de la siguiente forma:

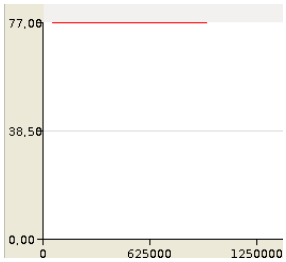
semilla	Valor	Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
11	última instancia	84.6985	77.049257456157	77.05770695518144
11	media	84.67	77.01	77.01

Tabla 2: Métricas de rendimiento del clasificador HoeffdingTree estacionario entrenado con un generador WafeformGenerator con seed 11

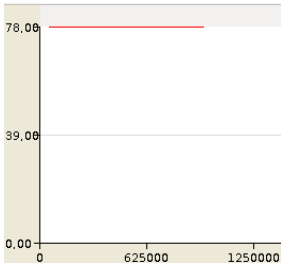
Repetimos esta experimentación estableciendo como semilla el valor 23 para el generador usado para el entrenamiento del clasificador:



(a) Accuracy



(b) Estadístico Kappa



(c) Kappa Temporal

Figura 3: Representación gráfica de la evolución de las métricas de Accuracy, Kappa y Kappa Temp del clasificador HoeffdingTree estacionario entrenado con un generador WaveformGenerator considerando una semilla aleatoria de 23.

Se observa un rendimiento similar al de la anterior experimentación. Las métricas con esta semilla resultan de la siguiente forma:

semilla	Valor	Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
23	última instancia	84.6502	76.97687332371794	76.98528838484097
23	media	84.63	76.95	76.95

Tabla 3: Métricas de rendimiento del clasificador HoeffdingTree estacionario entrenado con un generador WafeformGenerator con seed 23

En todos los experimentos, se obtienen unos resultados similares: El rendimiento del clasificador se mantiene casi constante a lo largo del flujo y alcanza un rendimiento alto.

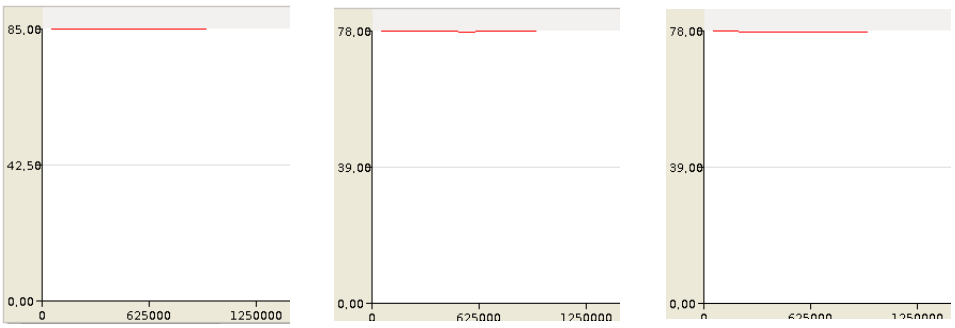
2.1.2. Repetir el paso anterior, sustituyendo el clasificador por HoeffdingTree adaptativo

Repetimos las experimentaciones anteriores usando un clasificador HoeffdingTree adaptativo capaz de adapatarse a los cambios de concepto (*HoeffdingAdaptativeTree*).

Lanzamos esta experimentación mediante el siguiente comando en el que se ha establecido este clasificador como modelo a entrenar:

```
EvaluateInterleavedTestThenTrain -l (LearnModel -l trees.HoeffdingAdaptiveTree
                                      -s (generators.WaveformGenerator -i 2)
                                      -m 1000000)
-s (generators.WaveformGenerator -i 4)
-i 1000000
```

Con este modelo, la evolución de las métricas de rendimiento a lo largo del flujo, se desarrollan de la siguiente forma:



(a) Accuracy (b) Estadístico Kappa (c) Kappa Temporal

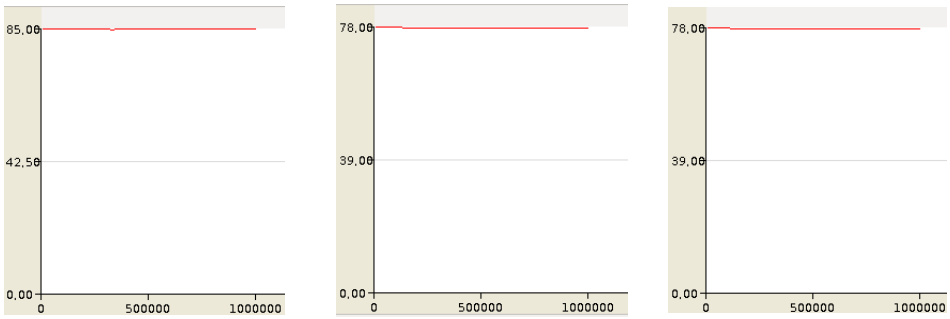
Figura 4: Representación gráfica de la evolución de las métricas de Accuracy, Kappa y Kappa Temp del clasificador HoeffdingAdaptiveTree a lo largo del flujo considerando una semilla aleatoria para el generador de 2.

Las métricas del clasificador al terminar el flujo y en media son las siguientes:

semilla	Medida	Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
2	última instancia	84.5884	76.8844644874789	76.89262859918796
2	media	84.55	76.82	76.82

Tabla 4: Métricas de rendimiento del clasificador HoeffdingAdaptiveTree con seed 2 en el generador de instancias

Se repite esta experimentación estableciendo un valor 11 para la semillas aleatoria del generador:



(a) Accuracy (b) Estadístico Kappa (c) Kappa Temporal

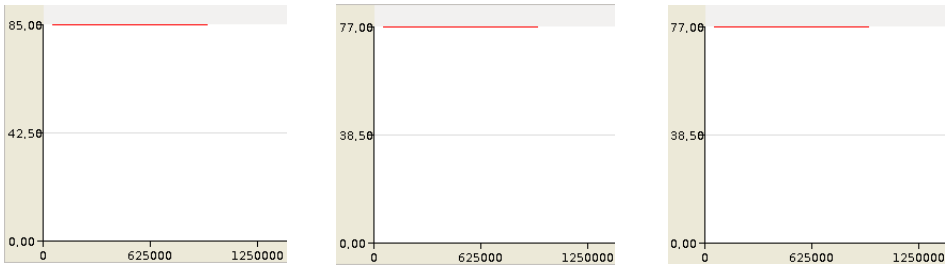
Figura 5: Representación gráfica de la evolución de las métricas de Accuracy, Kappa y Kappa Temp del clasificador HoeffdingAdaptiveTree a lo largo del flujo considerando una semilla aleatoria 11 para el generador.

Las métricas del clasificador al terminar el flujo son las siguientes:

semilla	Medida	Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
11	última instancia	84.5555	76.83488314068066	76.84330000779661
11	media	84.56	76.85	76.85

Tabla 5: Métricas de rendimiento del clasificador HoeffdingAdaptiveTree con seed 11 en el generador de instancias

Se realiza otra experimentación estableciendo un valor 23 para la semillas aleatoria del generador:



(a) Accuracy (b) Estadístico Kappa (c) Kappa Temporal

Figura 6: Representación gráfica de la evolución de las métricas de Accuracy, Kappa y Kappa Temp del clasificador HoeffdingAdaptiveTree a lo largo del flujo considerando una semilla aleatoria 23 para el generador.

Las métricas del clasificador al terminar el flujo son las siguientes:

semilla	Medida	Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
23	última instancia	84.5625	76.84551228237578	76.85379545277348
23	media	84.53	76.79	76.79

Tabla 6: Métricas de rendimiento del clasificador HoeffdingAdaptiveTree con seed 23 en el generador de instancias

Comparando estas experimentaciones con las realizadas con el modelo *HoeffdingTree* original, se aprecia que ambos clasificadores obtienen rendimientos muy similares en este flujo de datos.

2.1.3. Responda a la pregunta: ¿Cree que algún clasificador es mejor que el otro en este tipo de problemas? Razone su respuesta

En este problema, ambos clasificadores entrenan de forma estacionaria (*offline*) frente a un flujo de datos sin cambio de concepto, por lo que el clasificador *Hoeffding Adaptive Tree* no aplicaría la eliminación de ramas de árbol que se consideraran obsoletas y se ejecutaría, por lo tanto, de forma similar al clasificador *Hoeffding Tree* original, por consiguiente, el rendimiento de ambos clasificadores resultaría similar en este problema al no existir cambio de concepto.

2.2. Entrenamiento online

2.2.1. Entrenar un clasificador HoeffdingTree online, mediante el método InterleavedTest-Then-Train, sobre un total de 1.000.000 de instancias procedentes de un flujo obtenido por el generador WaveFormGenerator con semilla aleatoria igual a 2, con una frecuencia de muestreo igual a 10.000. Pruebe con otras semillas aleatorias para crear una población de resultados. Anotar los valores de porcentajes de aciertos en la clasificación y estadístico Kappa

Implementamos esta experimentación mediante el siguiente comando:

```
EvaluateInterleavedTestThenTrain -l trees.HoeffdingTree
-s (generators.WaveformGenerator -i 2)
-i 1000000 -f 10000
```

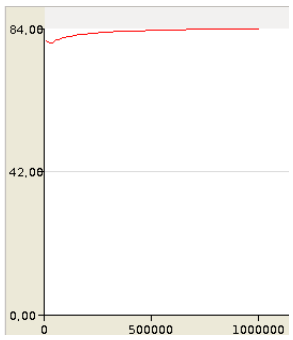
Esta experimentación también hace uso del método *EvaluateInterleaved-TestThenTrain* para realizar la evaluación y entrenamiento del modelo. En

esta experimentación se ha ajustado un número de instancias máximas a generar de 1.000.000 (parámetro `-i`) con una frecuencia de muestreo de 10.000 (parámetro `-f`).

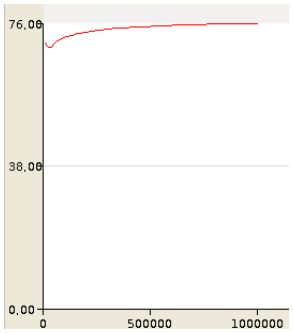
Se ha especificado también el generador *WaveformGenerator* con una semilla aleatoria para la generación de instancias de 2 (parámetro `-i` del generador).

Ejecutamos la experimentación y realizamos otra ejecución posterior con una semilla de 11 y otra con una semilla de 23. Para ambas experimentaciones, se obtienen las siguiente evolución en las métricas de *Accuracy*, *Kappa Static* y *Kappa Temporal Static*:

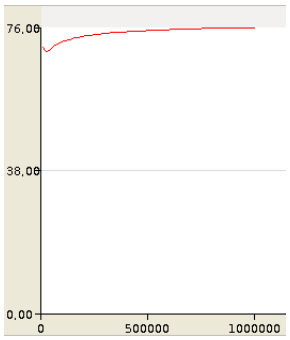
seed = 2



(a) Accuracy

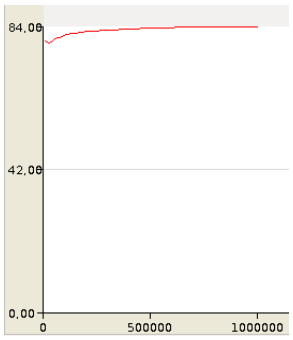


(b) Estadístico Kappa

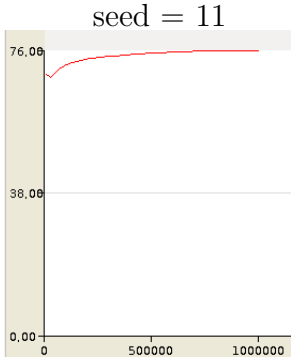


(c) Kappa Temporal

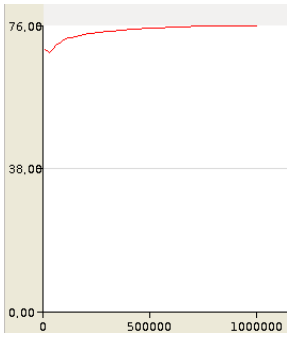
seed = 11



(d) Accuracy

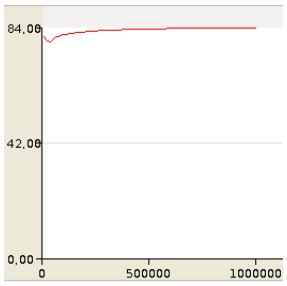


(e) Estadístico Kappa

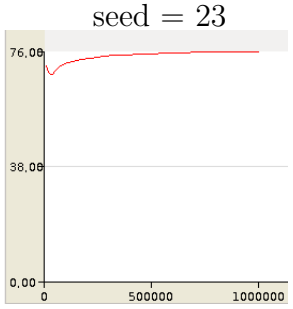


(f) Kappa Temporal

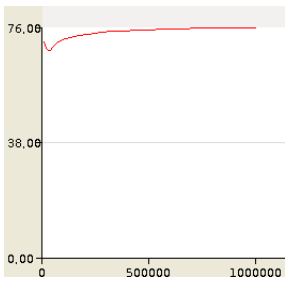
seed = 23



(g) Accuracy



(h) Estadístico Kappa



(i) Kappa Temporal

Figura 7: Representación gráfica de la evolución de las métricas de Accuracy, Kappa y Kappa Temp del clasificador HoeffdingTree entrenado online a lo largo del flujo considerando semillas aleatorias para el generador de instancias de 2, 11 y 23.

El valor de las métricas al acabar el flujo y en media son las siguientes:

semilla		Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
2	último instante	83.7851	75.67749801500895	75.67434171047262
2	media	82.85	74.27	74.28
11	último instante	83.7456	75.61783494622635	75.63095006071873
11	media	82.80	74.20	74.23
23	último instante	83.8462	75.76954694364466	75.78180869580862
23	media	82.89	74.34	74.33

Tabla 7: Métricas de rendimiento del clasificador HoeffdingTree online con seed 2, 11 y 23

Se observa que a lo largo del flujo, el rendimiento del clasificador se mantiene constante: En las primeras instancias del flujo, se observa un crecimiento creciente del rendimiento hasta lograr la convergencia en la cual, el rendimiento se mantiene constante.

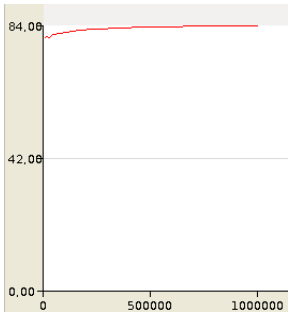
2.2.2. Repetir el paso anterior, sustituyendo el clasificador por HoeffdingTree adaptativo

Se repite la anterior experimentación con el clasificador *HoeffdingAdaptiveTree*, por lo que se edita el comando anterior modificando el clasificador. El comando quedaría de la siguiente forma:

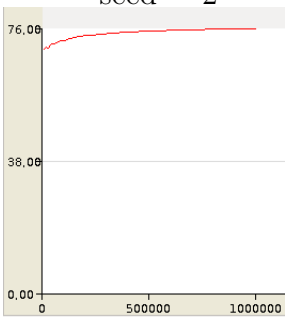
```
EvaluateInterleavedTestThenTrain -l trees.HoeffdingAdaptiveTree
-s (generators.WaveformGenerator -i 2)
-i 1000000 -f 10000
```

Ejecutamos esta experimentación con valores de semilla de 2, 11 y 23 obtiendo la siguiente evolución en las métricas a lo largo del flujo:

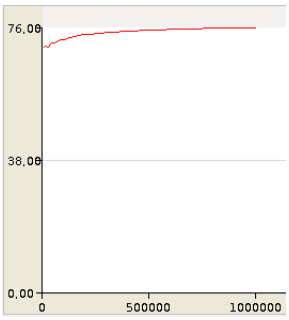
seed = 2



(a) Accuracy

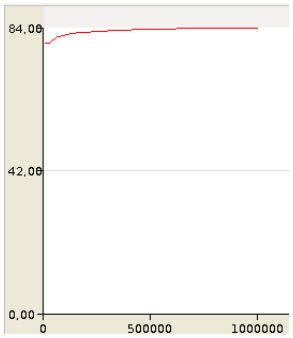


(b) Estadístico Kappa

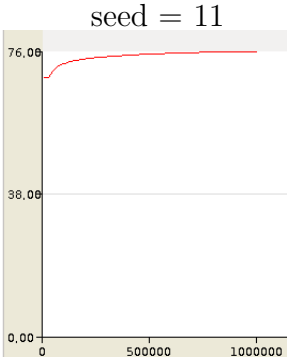


(c) Kappa Temporal

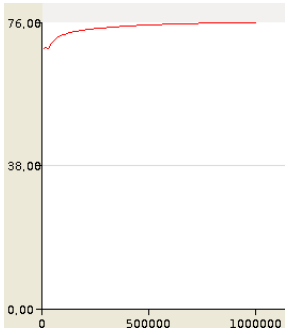
seed = 11



(d) Accuracy

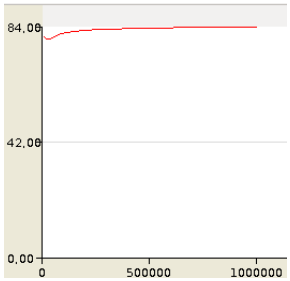


(e) Estadístico Kappa

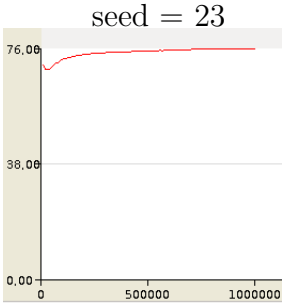


(f) Kappa Temporal

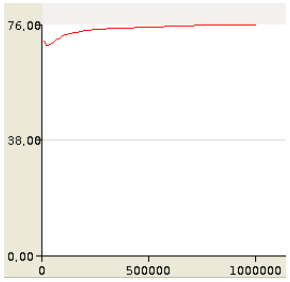
seed = 23



(g) Accuracy



(h) Estadístico Kappa



(i) Kappa Temporal

Figura 8: Representación gráfica de la evolución de las métricas de Accuracy, Kappa y Kappa Temp del clasificador HoeffdingAdaptiveTree entrenado online a lo largo del flujo considerando una semilla aleatoria para la generación de instancias de 2, 11 y 23.

El valor de las métricas al acabar el flujo y en media son las siguientes:

semilla	Medida	Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
2	último instante	83.7313	75.5967623000447	75.59363073377979
2	media	82.87	74.31	74.31
11	último instante	83.7456	75.61783494622635	75.63095006071873
11	media	82.88	74.32	74.35
23	último instante	83.6508	75.47646917494608	75.48886000380803
23	media	82.74	74.11	74.11

Tabla 8: Métricas de rendimiento del clasificador HoeffdingAdaptiveTree online con seed 2, con seed 11 y con seed 23

El rendimiento con este clasificador es similar al del clasificador *HoeffdingTree* online que no implementa ningún mecanismo de adaptación.

2.2.3. Responda a la pregunta: ¿Cree que algún clasificador es mejor que el otro en este tipo de problemas? Razone su respuesta

En este caso, y al igual que en la experimentación de la sección 2.1, al entrenar también sobre un flujo de datos sin cambio de concepto, ambos clasificadores se ejecutan de forma similar, por lo que su rendimiento sobre el flujo de datos es también similar.

2.3. Entrenamiento online en datos con concept drift.

2.3.1. Entrenar un clasificador HoeffdingTree online, mediante el método InterleavedTest-Then-Train, sobre un total de 2.000.000 de instancias muestreadas con una frecuencia de 100.000, sobre datos procedentes de un generador de flujos RandomRBFGeneratorDrift, con semilla aleatorio igual a 1 para generación demodelos y de instancias, generando 2 clases, 7 atributos, 3 centroides en elmodelo, drift en todos los centroides y velocidad de cambio igual a 0.001. Pruebe con otras semillas aleatorias. Anotar los valores de porcentajes de aciertos en la clasificación y estadístico Kappa. Compruebe la evolución de lacurva de aciertos en la GUI de MOA.

Programamos esta experimentación por medio del siguiente comando:

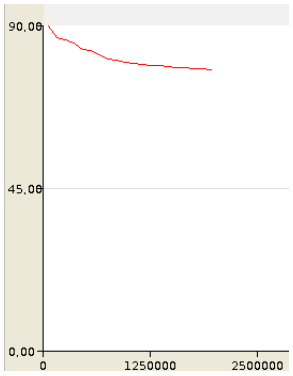
```
EvaluateInterleavedTestThenTrain -l trees.HoeffdingTree
-s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -a 7 -c 2 -n 3 -i 1 -r 1)
-i 2000000 -f 100000
```

Mediante el anterior comando, se entrena de forma online un clasificador *HoeffdingTree* con un flujo de un máximo de 2.000.000 instancias (parámetro `-i`) con una frecuencia de muestreo de 100.000 instancias (parámetro `-f`) generadas mediante un generador *RandomRBFGeneratorDrift* (parámetro `-s`).

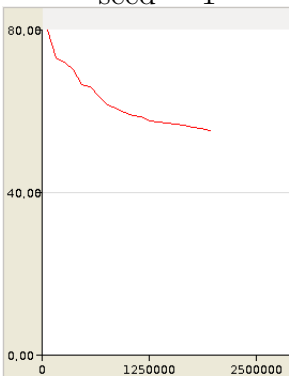
El generador *RandomRBFGeneratorDrift* generaría instancias con 7 atributos (parámetro `-a` del generador) pertenecientes a 2 clases (parámetro `-c` del generador) con 3 centroides (parámetro `-k`), un cambio de concepto por cada centroide (lo que resulta en 3 cambios de concepto que se especifican por medio del parámetro `-n`) y una densidad de 0.001 (especificada mediante el parámetro `-s`). Por último, se ha especificado una semilla de valor 1 para el generador y para las instancias generadas (parámetros `-r` y `-i` respectivamente).

Ejecutamos esta experimentación con valores de semilla de 1, 11 y 23 obteniendo la siguiente evolución en las métricas a lo largo del flujo:

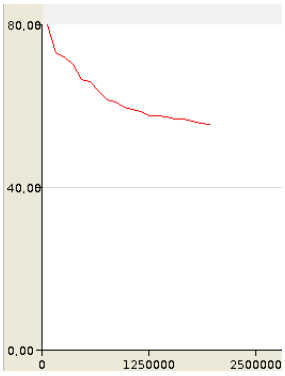
seed = 1



(a) Accuracy

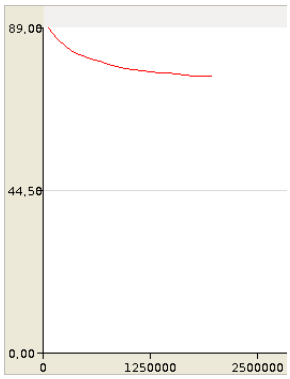


(b) Estadístico Kappa

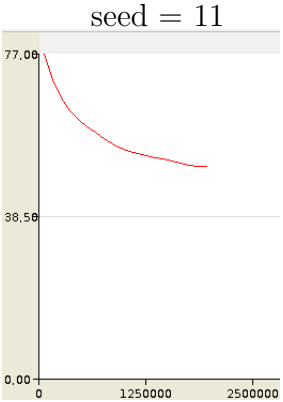


(c) Kappa Temporal

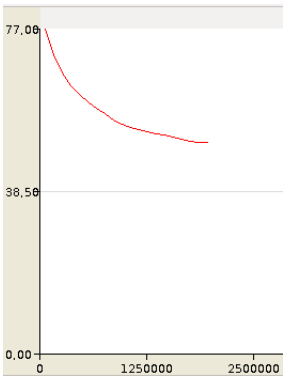
seed = 11



(d) Accuracy

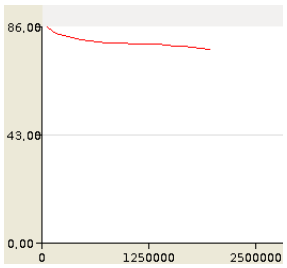


(e) Estadístico Kappa

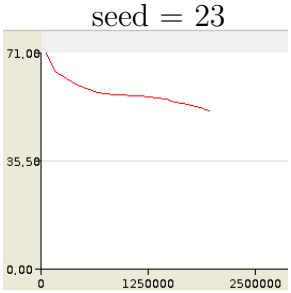


(f) Kappa Temporal

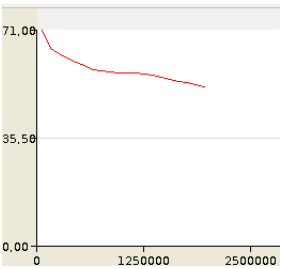
seed = 23



(g) Accuracy



(h) Estadístico Kappa



(i) Kappa Temporal

Figura 9: Representación gráfica de la evolución de las métricas de Accuracy, Kappa y Kappa Temp del clasificador HoeffdingTree entrenado online a lo largo del flujo generado con RandomRBFGenerator Drift considerando valores de semilla 1, 11 y 23 tanto para el generador como para las instancias generadas.

El valor de las métricas al acabar el flujo y en media son las siguientes:

semilla	Valor	Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
1	último instante	77.45465	54.77595805976361	54.852866082603256
1	media	80.86	61.65	61.66
11	último instante	74.93125	49.80766643354869	49.62234974121639
11	media	78.38	56.65	56.55
23	último instante	77.00125	51.3090122358794	51.792878373109865
23	media	79.75	57.10	57.56

Tabla 9: Métricas de rendimiento del clasificador HoeffdingTree entrenado online a lo largo del flujo generado con RandomRBFGenerator Drift considerando un seed 1, de 11 y 23

Observamos como el rendimiento del clasificador disminuye a lo largo del flujo hasta converger en su valor mínimo evaluado en el último instante.

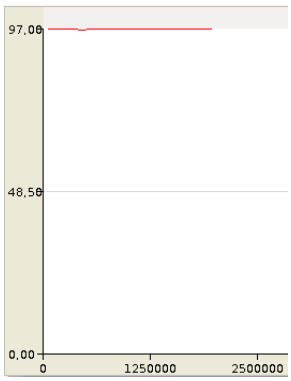
2.3.2. Repetir el paso anterior, sustituyendo el clasificador por HoeffdingTree adaptativo

Repetimos la anterior experimentación haciendo uso de un clasificador *HoeffdingAdaptiveTree* para lo cual modificamos el comando usado en el anterior apartado de la siguiente forma:

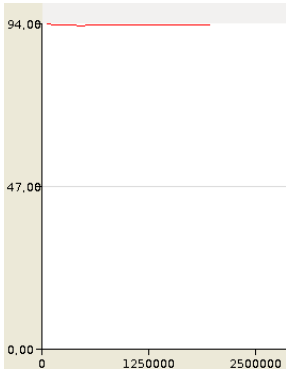
```
EvaluateInterleavedTestThenTrain -l trees.HoeffdingAdaptiveTree
-s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -a 7 -c 2 -n 3 -i 1 -r 1)
-i 2000000 -f 100000
```

Ejecutamos esta experimentación con valores de semilla de 1, de 11 y de 23 tanto en el generador de instancia como en las intenaicas generados, obteniendo la siguiente evolución en las métricas de rendimiento:

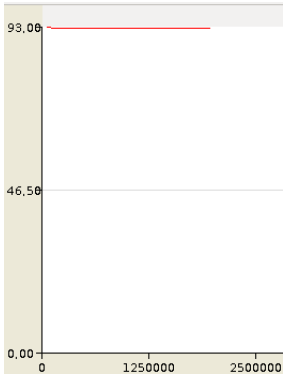
seed = 1



(a) Accuracy

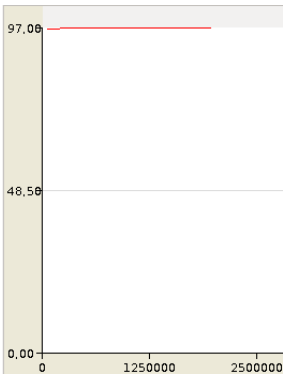


(b) Estadístico Kappa

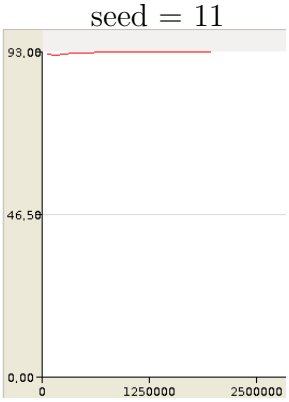


(c) Kappa Temporal

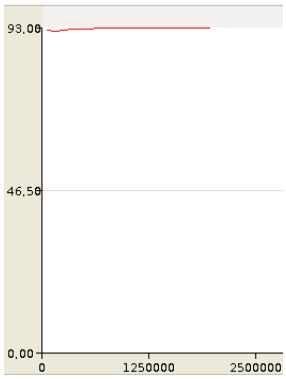
seed = 11



(d) Accuracy

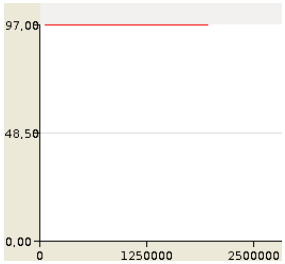


(e) Estadístico Kappa

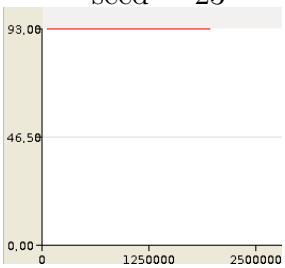


(f) Kappa Temporal

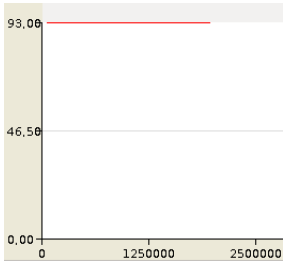
seed = 23



(g) Accuracy



(h) Estadístico Kappa



(i) Kappa Temporal

Figura 10: Representación gráfica de la evolución de las métricas de Accuracy, Kappa y Kappa Temp del clasificador HoeffdingAdaptiveTree entrenado online a lo largo del flujo generado con RandomRBFGenerator Drift, considerando semillas de 1, de 11 y de 23 tanto para el generador como para las instancias generadas.

El valor de las métricas al acabar el flujo y en media son las siguientes:

semilla	Valor	Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
1	último instante	96.24925	92.49138450975714	92.48911138923654
1	media	96.27	92.53	92.53
11	último instante	96.02340000000001	91.98694275549316	92.00870549911428
11	media	95.94	91.82	91.85
23	último instante	96.1635	91.88654109665276	91.95840547327295
23	media	96.16	91.89	91.96

Tabla 10: Métricas de rendimiento del clasificador HoeffdingAdaptiveTree online entrenado online a lo largo del flujo generado con RandomRBFGenerator Drift, considerando una semilla 1, 11 y 23 tanto para el generador como para las instancias generadas

Los resultados de este clasificador nos muestran una tendencia completamente diferente al del clasificador *HoeffdingTree* original: La evolución de las métricas se mantiene más o menos constante a lo largo del flujo y con unos valores muy óptimos, lo que nos lleva a comprobar que el método adaptativo puede ajustarse mejor a este flujo con cambios de concepto que el método original.

2.3.3. Responda a la pregunta: ¿Cree que algún clasificador es mejor que el otro en este tipo de problemas? Razone su respuesta

El flujo tratado en esta sección presenta cambios de concepto: el clasificador *Hoeffding Adaptive Tree* al implementar un mecanismo para eliminar las ramas que ya no ajustan a los datos tras el cambio de concepto, puede ajustarse mejor a la nueva dinámica de los datos y "olvidar" el ajuste a los anteriores datos así como el sesgo que introducen para la clasificación de los nuevos datos, lo cual no es posible para el clasificador *Hoeffding Tree* para el que la clasificación sobre los datos tras los cambios de concepto se ve fuertemente sesgada por al ajuste a los datos anteriores al cambio de concepto.

Por consiguiente y por lo general, en este tipo de problema con cambio de concepto, el clasificador *Hoeffding Adaptive Tree*, ofrecerá mejores rendimientos que el clasificador *Hoeffding Tree* original.

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

2.4.1. Repita la experimentación del apartado anterior, cambiando el método de evaluación "Interleaved Test-Then-Train" por el método de evaluación"Prequential", con una ventana deslizante de tamaño 1.000

Repetimos el experimento con el clasificador *HoeffdingTree* online con un valor de semilla de 1, pero modificamos el comando cambiando el evaluador de *EvaluateInterleavedTestThenTrain* por *EvaluatePrequential*:

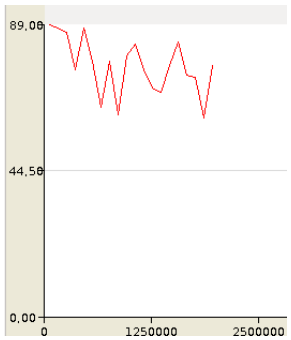
```
EvaluatePrequential -l trees.HoeffdingTree
-s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -a 7 -c 2 -n 3 -i 1 -r 1)
-i 2000000 -f 100000 -w 1000
```

En este comando se hace uso del evaluador *EvaluatePrequential* tomando el mismo clasificador y el mismo generador con las mismas configuraciones que en el apartado anterior.

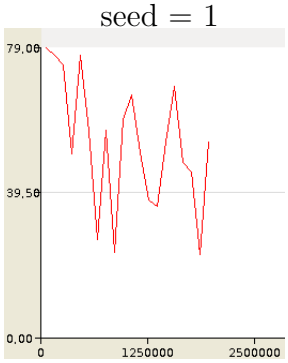
En esta experimentación, se mantiene un número de instancias máximas a generas de 2.000.000 (parámetro `-i`) con una frecuencia de muestreo de 100.000 (parámetro `-f`) y se establece un tamaño de ventana deslizante de 1.000 instancias (parámetro `-w`).

Ejecutamos esta experimentación con el valor de semilla 1 y la repetimos para un valor de semilla 11.

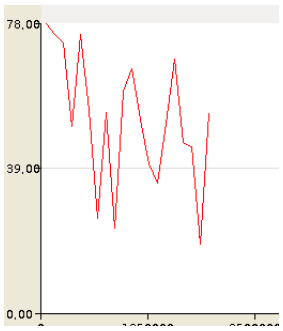
seed = 1



(a) Accuracy

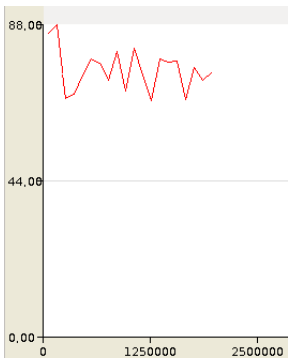


(b) Estadístico Kappa

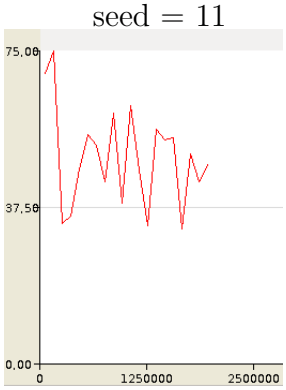


(c) Kappa Temporal

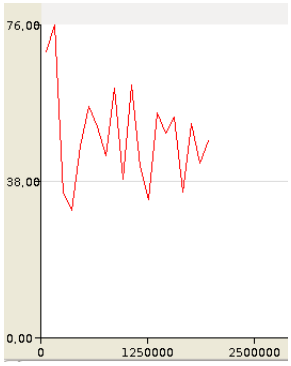
seed = 11



(d) Accuracy

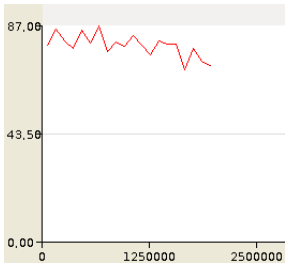


(e) Estadístico Kappa

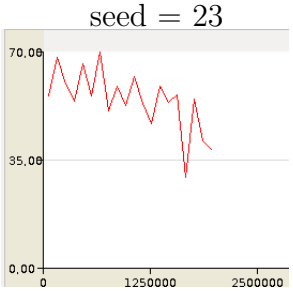


(f) Kappa Temporal

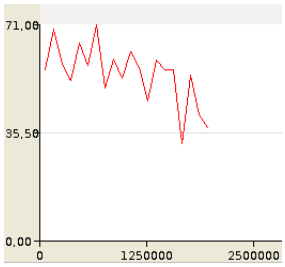
seed = 23



(g) Accuracy



(h) Estadístico Kappa



(i) Kappa Temporal

Figura 11: Representación gráfica de la evolución de las métricas de Accuracy, Kappa y Kappa Temp del clasificador HoeffdingTree entrenado online a lo largo del flujo generado con RandomRBFGenerator Drift, considerando semillas de 1, 11 y 23 tanto para el generador como para las instancias generadas y usando el método de evaluación prequencial.

El valor de las métricas al acabar el flujo y en media son las siguientes:

semilla	Valor	Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
1	último instante	76.6	52.55878453173087	53.663366336633665
1	media	77.40	54.54	54.65
11	último instante	73.6	47.55244755244755	47.41035856573705
11	media	75.23	50.43	50.13
23	último instante	70.0	38.16982687551525	36.84210526315788
23	media	78.07	53.78	54.29

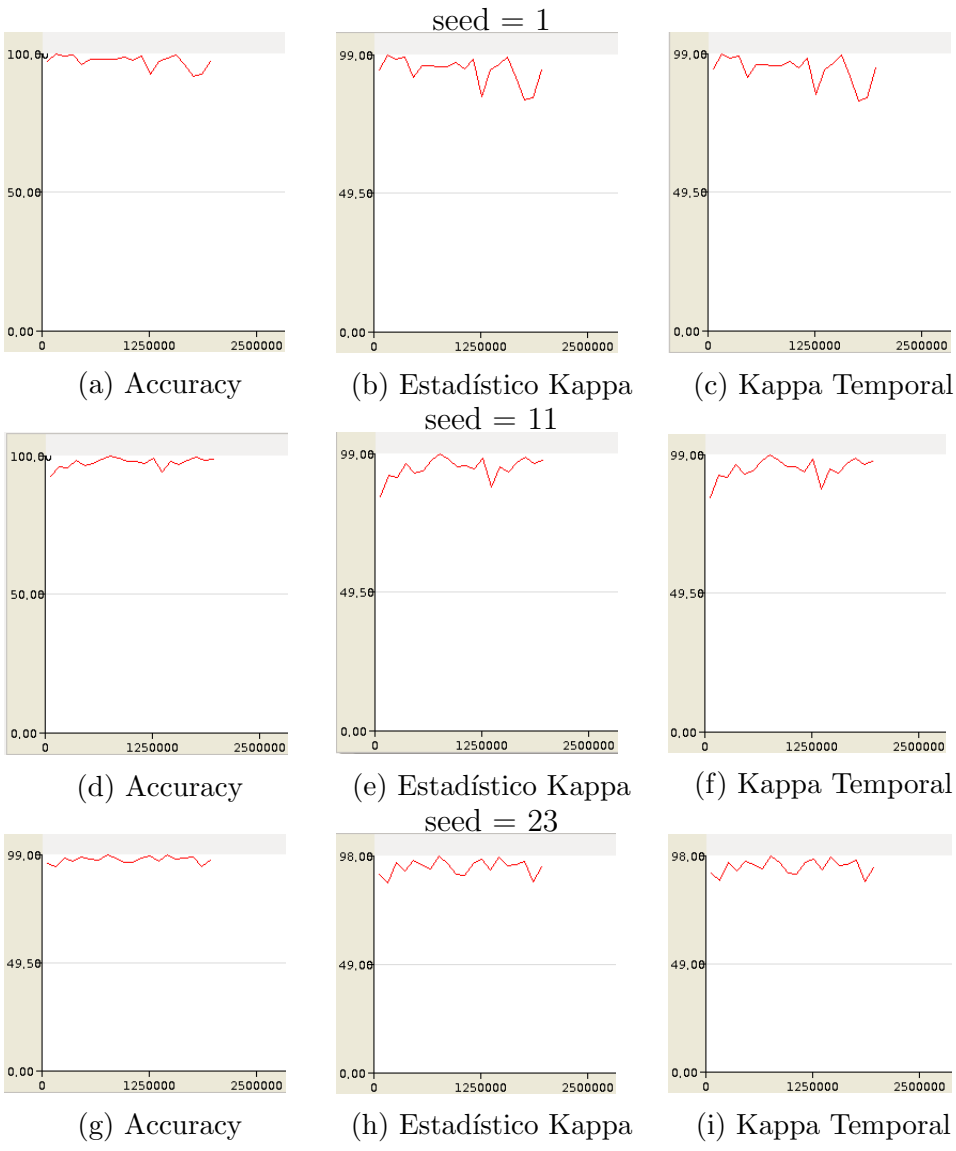
Tabla 11: Métricas de rendimiento del clasificador HoeffdingAdaptiveTree online entrenado online a lo largo del flujo generado con RandomRBFGenerator Drift, considerando semillas de 1, 11 y 23 tanto para el generador como para las instancias generadas y usando el método de evaluación prequencial

Observamos que el rendimiento de este clasificador sufre grandes oscilaciones a lo largo del flujo al tiempo que presenta una tendencia decreciente en todas sus métricas de rendimiento.

Por último, repetimos esta experimentación con *HoefdingAdaptiveTree*, para lo cual modificamos en el anterior comando el clasificador usado por este último:

```
EvaluatePrequential -l trees.HoeffdingAdaptiveTree
-s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -a 7 -c 2 -n 3 -i 1 -r 1)
-i 2000000 -f 100000 -w 1000
```

Tomando una semilla de 1, de 11 y de 23 tanto para el generador como en la generación de instancias, las métricas resultan del siguiente modo:



(a) Accuracy (b) Estadístico Kappa (c) Kappa Temporal

(d) Accuracy (e) Estadístico Kappa (f) Kappa Temporal

(g) Accuracy (h) Estadístico Kappa (i) Kappa Temporal

Figura 12: Representación gráfica de la evolución de las métricas de Accuracy, Kappa y Kappa Temp del clasificador HoeffdingAdaptiveTree entrenado online a lo largo del flujo generado con RandomRBFGenerator Drift, considerando semillas de valor 1, 11 y 23 tanto para el generador como para las instancias generadas y usando el método de evaluación prequencial.

El valor de las métricas al acabar el flujo y en media son las siguientes:

semilla	Valor	Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
1	último instante	96.6	93.19071213134715	93.26732673267327
1	media	96.48	93.19	93.27
11	último instancia	98.1	96.14910659272951	96.21513944223106
11	media	96.84	93.62	93.61
23	último instancia	96.2	92.07011686143571	92.0
23	media	96.25	92.03	92.21

Tabla 12: Métricas de rendimiento del clasificador HoeffdingAdaptiveTree online entrenado online a lo largo del flujo generado con RandomRBFGenerator Drift, considerando semillas de valor 1, de 11 y de 23 tanto para el generador como para las instancias generadas y usando el método de evaluación prequencial

Aunque en ambos clasificadores se observan oscilaciones en las métricas de rendimiento, en este clasificador las oscilaciones son mucho más leves y la tendencia que se aprecia es creciente. Las métricas de rendimiento obtenidas por este modelo ofrece valores muy superiores a los del modelo *HoeffdingTree*.

2.4.2. ¿Qué efecto se nota en ambos clasificadores? ¿A qué es debido? Justifique los cambios relevantes en los resultados de los clasificadores

Se aprecian efectos oscilatorios en ambos clasificadores. El método de evaluación prequencial, a diferencia de *InterleavedTest-Then-Train*, evalúa el rendimiento sobre las instancias más recientes que se incluyen en la ventana utilizada y no sobre todas las instancias, lo que acentúa aún más las oscilaciones y pérdida de rendimiento de *HoeffdingTree* debido a su incapacidad para desechar los ajustes previos a los cambios de concepto, que provocaría la aparición de picos bajos y bruscos de rendimiento en las instancias inmediatamente posteriores a cada cambio de concepto.

Al igual que se detalló en la anterior sección, los ajustes a los datos previos a cada cambio de concepto sesgarían fuertemente la clasificación sobre las instancias posteriores y ello llevaría a la pérdida de rendimiento progresivo de este clasificador y de ahí, la tendencia decreciente del rendimiento.

Por el contrario, el clasificador *Hoeffding Adaptive Tree* que sí implementa mecanismos para desajustar los datos anteriores a los cambios de concepto y ajustar de forma óptima a los datos posteriores, en las instancias inmedietamente posteriores a los cambios de concepto, el rendimiento evaluado sobre la ventana con estas instancias no presenta un descenso tan bruco puesto que

el modelo se adapta rápidamente a la nueva dinámica de los datos.

Dado que el modelo sí ajusta correctamente a los datos tras el cambio de concepto, el rendimiento del modelo muestra tendencia creciente.

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

2.5.1. Repita la experimentación del apartado 2.3, cambiando el modelo (learner) a un clasificador simple basado en reemplazar el clasificador actual cuando se detecta un cambio de concepto (SingleClassifierDrift). Como detector de cambio deconcepto, usar el método DDM con sus parámetros por defecto. Como modelo aaprender, usar un clasificador HoeffdingTree

Repetimos la experimentación del apartado 2.3 usando el mismo flujo para entrenar y evaluar los modelos, pero en esta experimentación, configuraremos el aprendizaje para que se entrene un nuevo modelo de *HoeffdingTree* original cada vez que se detecte un cambio de concepto con el método *DDM*.

Configuramos esta experimentación mediante el siguiente comando:

```
EvaluatePrequential -l (moa.classifiers.drift.SingleClassifierDrift
                        -l trees.HoeffdingTree -d DDM)
-s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -a 7 -c 2 -n 3 -i 1 -r 1)
-i 2000000 -f 100000 -w 1000
```

Para elaborar los modelos se usa el método *SingleClassifierDrift* (parámetro `-l` del evaluador) y se especifica la elaboración de modelos basados en *HoeffdingTree* (parámetro `-l` de la utilidad *SingleClassifierDrift*) con el método *DDM* con sus parámetros por defecto (parámetro `-d` de la utilidad *SingleClassifierDrift*) para la detección de los cambios de concepto.

Considerando semillas de 1, 11 y 23 tanto en el generador de instancias como en las instancias generadas, obtenemos el siguiente rendimiento:

semilla	Valor	Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
1	último instante	97.7	95.38397315479344	95.44554455445544
1	media	97.50	94.99	94.93
11	último instancia	98.1	96.14034517502519	96.21513944223106
11	media	97.12	94.19	94.19
23	último instancia	99.2	98.33194328607172	98.3157894736842
23	media	97.05	93.71	93.85

Tabla 13: Métricas de rendimiento del clasificador HoeffdingTree con un método de reentramiento tras cada cambio de concepto a lo largo del flujo generado con RandomRBFGenerator Drift, considerando una semilla 1, de 11 y de 23 tanto para el generador como para las instancias generadas y usando el método de evaluación prequencial

2.5.2. Repita el paso anterior cambiando el clasificador HoeffdingTree por un clasificador HoeffdingTree adaptativo

Se repite esta experimentación con el clasificador *HoeffdingAdaptiveTree* modificando el parámetro `-l` de *SingleClassifierDrift*, por lo que quedaría del siguiente modo:

```
EvaluatePrequential -l (moa.classifiers.drift.SingleClassifierDrift
                        -l trees.HoeffdingAdaptiveTree -d DDM)
-s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -a 7 -c 2 -n 3 -i 1 -r 1)
-i 2000000 -f 100000 -w 1000
```

semilla	Valor	Accuracy (%)	Est. Kappa (%)	Est. Kappa Temporal (%)
1	último instante	97.3	94.57983699361625	94.65346534653465
1	media	96.94	93.87	93.76
11	último instancia	97.7	95.32565928527298	95.41832669322709
11	media	96.99	93.93	93.89
23	último instancia	99.0	97.9235880398671	97.89473684210526
23	media	96.70	93.0	93.15

Tabla 14: Métricas de rendimiento del clasificador HoeffdingTree con un método de reentramiento tras cada cambio de concepto a lo largo del flujo generado con RandomRBFGenerator Drift, considerando una semilla 1, de 11 y de 23 tanto para el generador como para las instancias generadas y usando el método de evaluación prequencial

2.5.3. Responda a la siguiente pregunta: ¿Qué diferencias se producen entre los métodos de los apartados 2.3, 2.4 y 2.5? Explique similitudes y diferencias entre las diferentes metodologías, y discuta los resultados obtenidos por cada una de ellas en el flujo de datos propuesto

En la sección 2.3, se someten los clasificadores *HoeffdingTree* y *HoefdingAdaptiveTree* a un flujo con cambio de concepto usando como método de evaluación y entrenamiento *InterleavedTest-Then-Train*, mientras que en la sección 2.4, se usa como método de evaluación y entrenamiento el método prequencial que, a diferencia de *InterleavedTest-Then-Train* realiza la evaluación sobre una ventana que incluye únicamente las instancias más recientes, por lo que la evaluación se realiza únicamente teniendo en cuenta las instancias más recientes. Por su parte, el entrenamiento de los clasificadores se produce de forma similar con ambos métodos de evaluación y lo único que difiere es la evaluación y los valores de las métricas evaluadas.

Por último, con respecto a la sección 2.4, en la sección 2.5 se entrenan clasificadores basados en *HoeffdingTree* y *HoefdingAdaptiveTree* que hacen uso del método *DDM* para la deteción del *Concept Drift* y reentrenar de nuevo el clasificador con los datos posteriores al cambio de concepto, lo que resulta en una mejora significativa en el rendimiento del clasificador *HoeffdingTree*, que al reentrenar de nuevo tras un cambio de concepto, elimina el ajuste realizado a datos anteriores al cambio de concepto y que introducían un sesgo significante para la clasificación de las instancias posteriores, permitiendole así obtener resultados óptimos en los datos posteriores a cada cambio de concepto y, a lo largo del flujo como se ha podido comprobar en los experimentos.

No sucede lo mismo con el método *HoeffdingAdaptiveTree*, para el que el uso de *DDM* para detectar los cambios de concepto y reentrenar de nuevo el modelo cuando se produce dicho cambio de concepto no aporta ninguna mejora significativa sobre su rendimiento original dado que *HoeffdingAdaptiveTree* ya implementaba de por sí un mecanismo para desechar las ramas que no ajustan a los datos posteriores a cada cambio de concepto.