Universitat Oberta de Catalunya (UOC)

Máster Universitario en Ciencia de Datos (*Data Science*)

TRABAJO FINAL DE MÁSTER

Área: 2

Estudio de la presencia del fenómeno *Concept Drift* en el tratamiento en tiempo real del dataset 3W de Petrobras

---

Autor: Baltasar Boix Rita

Tutor: Jesús López Lobo

Profesor: Jordi Casas Roma

Tarragona, 19 de enero de 2023

Créditos/Copyright

Dataset 3W de Petrobras [31] en github sujeta a una licencia Apache 2.0:

> **"Este es el primer repositorio de Petrobras en GitHub. Apoya el proyecto 3W y promueve la experimentación de enfoques y algoritmos basados en *Machine Learning* para problemas específicos relacionados con eventos indeseables que ocurren en pozos de petróleo en alta mar."**

FICHA DEL TRABAJO FINAL

Título del trabajo:	Estudio de la presencia del fenómeno *Concept Drift* en el tratamiento en tiempo real del dataset 3W de Petrobras
Nombre del autor:	Baltasar Boix Rita
Nombre del colaborador docente:	Jesús López Lobo
Nombre del PRA:	Jordi Casa Roma
Fecha de entrega (mm/aaaa):	01/2023
Titulación o programa:	Máster Universitario en Ciencia de Datos
Área del Trabajo Final:	TFM Área 2
Idioma del trabajo:	Español
Palabras clave	Stream Learning, Concept Drift, Fault detection and diagnosis, Oil well monitoring.

Cita

Learn from the mistakes of others. You can't live long enough to make them all yourself.

— Eleanor Roosevelt

Agradecimientos

A mi familia.

A Txus, mi tutor, por su paciencia.

Abstract/Resumen

La temprana detección de eventos indeseables en pozos de petróleo y gas puede ayudar a prevenir accidentes con afectación a las personas y al medio ambiente además de reducir pérdidas de producción y costes de mantenimiento. Se ha utilizado el dataset 3W de Petrobras que contiene series temporales etiquetadas por expertos en condiciones normales de operación y en 8 tipos de situaciones indeseables. Se ha emulado la recepción de datos en tiempo real y se han tratado con modelos de *stream learning*. Se ha demostrado que es posible llevar a cabo una predicción en tiempo real de los eventos anómalos. Asimismo, se ha evaluado la presencia del fenómeno concept drift durante el proceso de aprendizaje con diferentes métodos a partir de los datos reales del dataset.

The early detection of undesirable events in oil and gas wells can help to prevent accidents with affectation to people and the environment in addition to reducing production losses and maintenance costs. The 3W dataset of Petrobras have been used. It contains time series labelled by experts in normal and in 8 types of undesirable situations. The reception of data in real time has been emulated and treated with *stream learning* models. It has been demonstrated that it is possible to predict anomalous events in real-time. Concept drift phenomenon has been evaluated during the learning process with different methods applied to the dataset real data.

Palabras clave: Stream Learning, Concept Drift, Fault detection and diagnosis, Oil well monitoring.

Índice general

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Descripción general del problema

Con el descubrimiento de grandes reservas de crudo en en el presalino brasileño[6], la producción de crudo en plataformas *offshore* se ha convertido en el principal recurso petrolífero de Brasil. La extracción de crudo en aguas profundas (Figura 1.1)presenta importantes retos tecnológicos y riesgos para la Seguridad y el Medio Ambiente. La detección temprana de situaciones anómalas ayuda a reducir estos riesgos.

Figura 1.1: Presalino de Brasil

.

El uso de ordenadores para el control de procesos complejos ha permitido grandes avances. Acciones de bajo nivel como la apertura y cerrado de válvulas, conocido como control regulatorio, que anteriormente era realizado por operadores humanos son ahora actuados automáticamente. Aun así, la respuesta ante situaciones anómalas del proceso sigue estando en buena medida en manos de los operadores. Esto implica la detección temprana de las situaciones anómalas, diagnosticando sus causas y tomando las acciones de control apropiadas para llevar el proceso de nuevo a un estado normal y seguro. Al conjunto de esta actividad se le ha denominado *Abnormal Event Management* (AEM), un componente clave del control supervisor[32]. Las estadísticas de Seguridad de la Industria de Proceso muestran que los grandes accidentes son infrecuentes pero cuando ocurren pueden transformar a todo un sector (*BP Deepwater Horizon Oil Spill* [26]). Los accidentes menores son más comunes pero también causan lesiones y enfermedades laborales o afectaciones medioambientales además de un gran coste acumulado. El primer paso en AEM es la automatización de la detección y diagnóstico de fallos en el proceso industrial. Para resolver este problema se han utilizado diferentes métodos. Venkatasubramanian *et al.* [32] clasifican estos métodos como los basados en modelos y los basados únicamente en datos históricos del proceso industrial.

Se utilizan técnicas de aprendizaje automático (*Machine Learning*, ML) para entrenar modelos que detectan y diagnostican situaciones anómalas a partir de datos en tiempo real. Podemos distinguir dos formas de entrenar los modelos[5]:

- Batch/Offline learning. Se selecciona un dataset de datos y se aplican técnicas de aprendizaje supervisado. Periódicamente se actualiza el dataset y se reentrena el modelo.
- Stream/Online learning. Los datos se adquieren secuencialmente en línea y se utilizan para actualizar el modelo. Los métodos de ML utilizados permiten el aprendizaje incremental (es ideal para los sistemas que reciben datos en tiempo real y necesitan poder adaptarse a condiciones que cambian rápidamente *Concept Drift*[29]).

El efecto *Concept Drift* consiste en el cambio que se puede producir en la distribución de los datos. Estos cambios pueden ser de varios tipos: súbitos, graduales, incrementales o cíclicos[21]. Si esto ocurre a menudo los modelos entrenados quedan obsoletos muy rápidamente. Se hace necesario detectar estos cambios en tiempo real y reentrenar los modelos.

En este TFM se van a utilizar métodos de *stream learning*[25] aplicados a las series temporales del dataset 3W de Petrobras[31].

El uso de *stream learning* en este caso presenta varios retos:

- El etiquetado lo realizan expertos en batch observando ventanas de tiempo de 5 minutos a 12 horas según la situación analizada (Tabla 3.1). O sea, la observación de un solo registro no contiene suficiente información para clasificarlo y será necesario analizar tendencias.

- El origen de los datos de cada serie temporal es diverso. Hay que pretratarlos para poder utilizar el conjunto.
- Los creadores del dataset[31] advierten que algunos de los tipos de eventos anómalos pueden sufrir de *concept drift*, cambios en el comportamiento de los datos que ocurren con el tiempo y deterioran el rendimiento de los modelos ajustados ([19]). Este va a ser el principal reto del TFM.

En definitiva, se va a tratar el dataset 3W como una fuente datos en línea que se someterá a pretratamiento y se ajustará a un modelo de aprendizaje incremental registro a registro que detecte si el proceso de extracción de crudo se encuentra en una situación estable o en una de las anómalas identificadas en el dataset. De la comparación entre de la etiqueta predicha por el modelo y su valor real se determinará si se está produciendo *concept drift* y es necesario reinicializar el modelo.

1.2. Motivación personal

En mis más de 30 años de experiencia en actividades de Producción de una gran instalación industrial me he enfrentado demasiadas veces a los problemas derivados de situaciones operatorias anómalas (averías de grandes máquinas, cortes de suministro eléctrico, roturas de calderas...). Expongo un caso real que viví en primera persona:

"En la sala de reuniones de Dirección estaban los responsables del negocio que tenían algo que decir sobre el incidente de la semana anterior en el que varios hornos de etileno habían sufrido temperaturas que provocaron que se fundieran los serpentines; el coste de reparación y el lucro cesante ascendía a varios millones de euros; afortunadamente no hubo que lamentar ningún accidentado. Entre todos los presentes acreditábamos más de 100 años de experiencia en la operación de Unidades de Proceso. El jefe de Planta expone la línea temporal de eventos y la evolución de las variables de Proceso más significativas. Se discute las decisiones tomadas por el operador de Control durante la emergencia. Se genera una discusión de pros y contras de las distintas opciones que nos lleva unas dos horas: ¡se podría haber hecho mejor y se habría evitado buena parte del coste!.
El operador de Control durante la emergencia estaba solo en su consola en una sala de control donde las alarmas no paraban de sonar y tuvo dos minutos para decidir."

Disponer de un programa de diagnóstico que ayude a saber que está pasando durante una emergencia en tiempo real nos puede ayudar a evitar las peores consecuencias de un incidente y añadimos una capa más al sistema de Seguridad[27]. El operador de Control también es una capa de Seguridad: ni la única ni la más importante.

1.3. Objetivos del TFM

En este trabajo se pretende demostrar que es posible llevar a cabo una predicción en tiempo real de los eventos anómalos producidos en la extracción de crudo en el dataset 3W de Petrobras [31].

- Objetivo principal. Contraste de hipótesis:
  - Hipótesis nula (H0): no se puede realizar una detección en tiempo real fiable de eventos anómalos durante el proceso de extracción de crudo.
  - Hipótesis alternativa (H1): es posible realizar una detección en tiempo real fiable de eventos anómalos durante el proceso de extracción de crudo.
- Objetivo secundario:
  - Considerar de forma adecuada la presencia del fenómeno *concept drift* durante el proceso de aprendizaje.

1.4. Descripción de la metodología

Las principales actividades en el TFM se pueden resumir:

- Análisis exploratorio y descriptivo del dataset
- Preprocesado de los datos. Reducción del volumen de datos por utilización de medias por minuto. Normalización de las variables para conjugar las series de varios orígenes.
- Aplicar algoritmos de aprendizaje incremental (*base learner*)

- Detección de *concept drift* con varias técnicas. No ha sido necesario introducir *drift* artificialmente[28].

Se utilizan principalmente métodos de *stream learning* del paquete River[25] recogiendo métricas de rendimiento de modelos en tiempo real .

1.5. Planificación

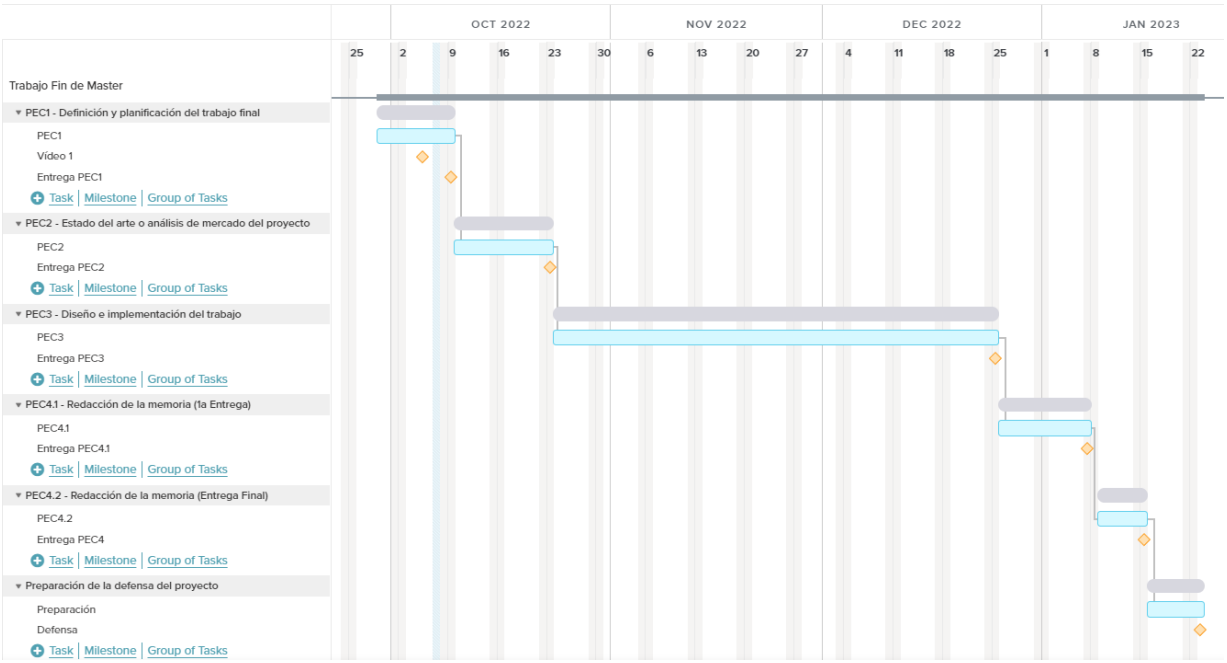


Figura 1.2: Planning del Trabajo Final de Master

.

1.6. Breve sumario de productos obtenidos

El TFM se ha desarrollado en *Jupyter Notebooks* donde se han programado rutinas que permiten filtrar el dataset por origen de los datos (reales, simulados y/o manuales) y evento anómalo a investigar; preprocesar los datos; seleccionar el modelo a ajustar; seleccionador el detector de *drift* y representar gráficamente los resultados.

Todo el proyecto se puede encontrar en https://github.com/BaltiBoix/3W_TFM. Una *fork* del *github* original del dataset 3W.

1.7. Impacto en sostenibilidad, ético-social y de diversidad

El propósito último de disponer de un modelo fiable de detección de situaciones anómalas en el proceso de extracción de crudo desde plataformas marinas es ayudar a que los operadores de la instalación puedan reaccionar a estas situaciones a tiempo de evitar incidentes/accidentes que afecten a la Seguridad de las personas, las instalaciones y el Medio ambiente. En jerga de Seguridad es añadir una capa preventiva más entre el peligro y el accidente.

En la industria del petróleo los accidentes graves son muy poco frecuentes pero tienen un coste económico y medioambiental muy elevado. En el análisis de los accidentes más graves tiene un papel relevante el factor humano. Las aplicaciones que mejoran la información de la que dispone el operador para tomar decisiones críticas reducen la probabilidad de error.

1.8. Breve descripción de los otros capítulos de la memoria

En el capítulo 2 **Estado del arte** se hace un breve repaso de bibliografía sobre *online learning* y *concept drift*. Asimismo se repasan los trabajos publicados que han utilizado el dataset 3W hasta la fecha.

En el capítulo 3 **Materiales y métodos** se desarrolla la metodología apuntada en la sección 1.4.

En el capítulo 4 **Experimentación y Resultados** se presentan los principales resultados gráficos de las ejecuciones de los modelos.

En el capítulo 5 **Conclusiones del TFM** se recoge el cumplimiento de los objetivos del TFM y se apuntan lineas de trabajo futuras.

Por último, se incluye un apéndice con un resumen de la aplicación de métodos offline al dataset 3W que, pese a no ser objetivo de este TFM, se muestran como referencia.

Capítulo 2

Estado del arte

2.1. Online Learning

Podemos distinguir dos formas de entrenar los modelos[5]:

- Batch/Offline learning. El modelo de ML se entrena utilizando todo el conjunto de datos que está disponible en un momento determinado. Una vez que tenemos un modelo que funciona bien en el conjunto de prueba, el modelo se envía a producción y, por lo tanto, finaliza el aprendizaje. Si con el paso del tiempo están disponibles nuevos datos es necesario actualizar el modelo. El modelo se entrena desde cero nuevamente utilizando las muestras de datos anteriores y las muestras de datos nuevos.

- Stream/Online learning. Los datos se transmiten (ya sea individualmente o en mini-lotes) al algoritmo de ML y actualizan el modelo. El aprendizaje en línea es ideal en situaciones en las que los datos se generan continuamente en el tiempo y necesitamos usar muestras de datos en tiempo real para construir un modelo de predicción. Un ejemplo típico de este caso es la predicción del mercado de valores.

El entorno para *Online Learning* impone unas restricciones computacionales con respecto a la configuración tradicional del *Batch Learning*[20]:

- Cada muestra se procesa solo una vez a su llegada, y los modelos deben poder procesar las muestras secuencialmente tan pronto como se reciben

- El procesamiento de cada muestra debe realizarse dentro del intervalo de tiempo entre muestras

- El algoritmo debe usar solo una cantidad de memoria preasignada y finita

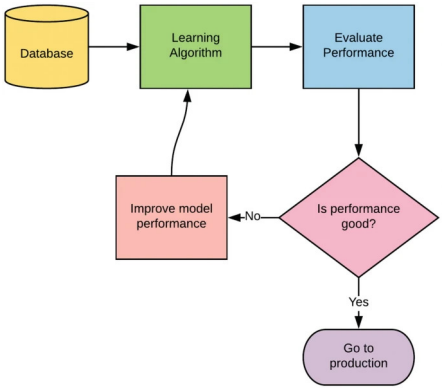


Figura 2.1: Batch Learning[5]

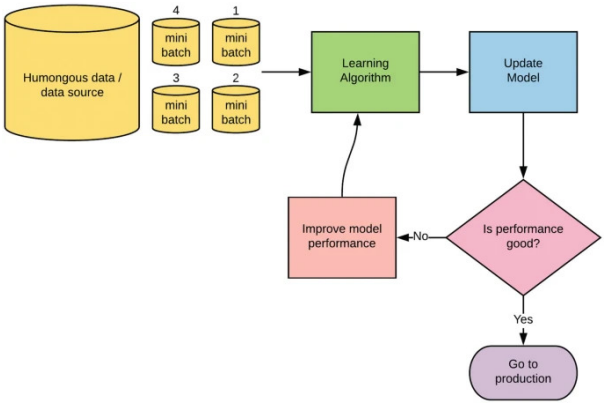


Figura 2.2: Online Learning[5]

- Tras cada exploración de datos, el modelo resultante debe ser válido para sus uso
- El modelo producido debe ser equivalente al que se construiría por *Batch Learning*.

Gomes *et al.*[16] hacen un repaso de los retos a los que se enfrenta el *Stream Learning* y las ventajas que tiene al enfrentarse al Big Data caracterizado por las 3 V's(Volumen, Variedad, Velocidad).

En este TFM se utilizan métodos de *stream learning* del módulo de Python River[25].

2.2. Concept Drift

Lu *et al.*[21] consideran que el *Concept drift* en el tiempo t puede definirse como el cambio en la probabilidad conjunta de X e y en el tiempo t. Puesto que la probabilidad conjunta $P_t(X, y)$ puede descomponerse en dos partes: $P_t(X, y) = P_t(X)P_t(y|X)$, *concept drift* puede provenir de tres orígenes (Figura 2.3):

- Origen I: $P_t(X) \neq P_{t+1}(X)$ mientras que $P_t(y|X) = P_{t+1}(y|X)$, esto es, varía $P_t(X)$ mientras que $P_t(y|X)$ no cambia. Ya que $P_t(X)$ no afecta la frontera de decisión, se considera *virtual drift*.
- Origen II: $P_t(y|X) \neq P_{t+1}(y|X)$ mientras que $P_t(X) = P_{t+1}(X)$. En este caso la frontera de decisión cambia y por tanto la precisión de los modelos empeora, se considera *Real Drift*.
- Origen III: mezcla del Origen I y II, o sea, $P_t(X) \neq P_{t+1}(X)$ y $P_t(y|X) \neq P_{t+1}(y|X)$.

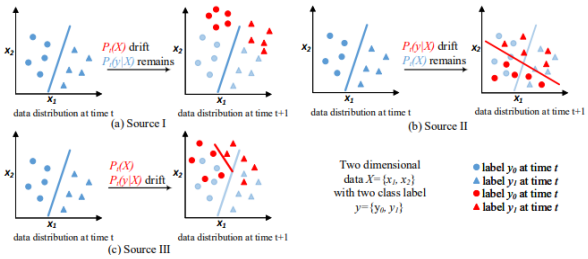


Figura 2.3: Orígenes de *Concept Drift*[21]

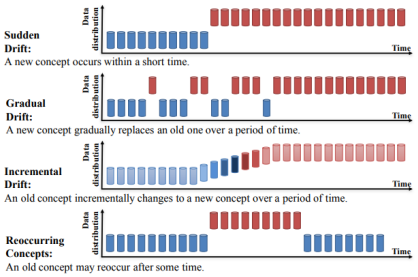


Figura 2.4: Tipos de *Concept Drift*[21]

Se pueden distinguir varios tipos de *Concept Drift*: súbitos, graduales, incrementales o cíclicos[21] (Figura 2.4).

Uno de los principales retos de los métodos para detectar *Concept Drift* es distinguirlo del ruido en el flujo de datos[29]. Lu *et al.*[21] proponen un marco general (Figura 2.5) para un detector de *Concept Drift* y hacen un repaso de diferentes algoritmos. Gama *et al.*[14] tratan el problema de adaptar los métodos de *Stream Learning* a la presencia de *Concept Drift* y proponen el esquema general de la Figura 2.6.

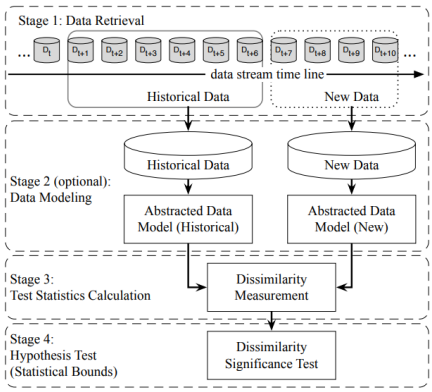


Figura 2.5: Esquema de un detector de *Concept Drift*[21]

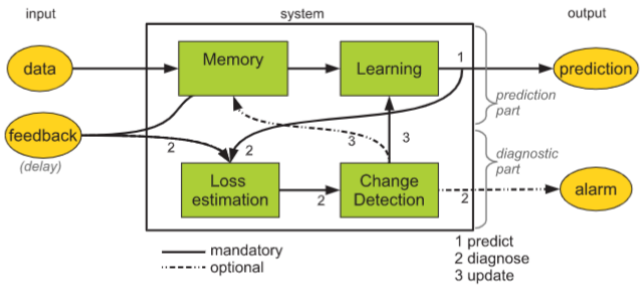


Figura 2.6: Esquema genérico para un *online adaptive learning algorithm*.[14]

Sethi *et al.*[28] exponen que la detección de *Concept Drift* se ha abordado tradicionalmente como una tarea supervisada, con datos etiquetados que se utilizan constantemente para validar el modelo aprendido. Aunque efectivas, estas técnicas no siempre son prácticas, ya que el etiquetado es una actividad difícil, costosa y que requiere mucho tiempo. Por otro lado, las técnicas de detección de cambios no supervisadas son poco fiables, ya que producen una gran cantidad de falsas alarmas. La ineficacia de las técnicas no supervisadas radica en la exclusión de las características del clasificador del proceso de detección. Hacen un repaso de la técnicas existentes y proponen un método no supervisado.

Hu *et al.*[17] y Gemaque *et al.*[15] hacen un repaso de la técnicas de detección no supervisadas de *Concept Drift*. Gemaque *et al.*[15] proponen la taxonomía de la figura 2.7.

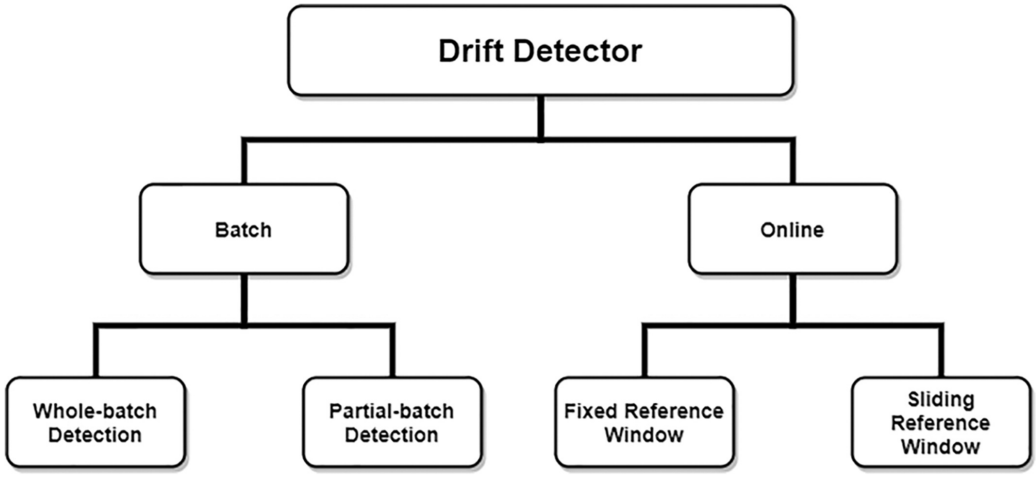


Figura 2.7: Taxonomía de métodos de detección no supervisada de *concept drift*[15]

Lobo *et al.*[22][20] investigan el uso de *Spiking Neural Networks* (SNN) en entornos de *online learning* y su comportamiento frente al *concept drift*.

En este TFM se aplica el esquema propuesto en la figura 2.6 con datos etiquetados.

2.3. Trabajos sobre *3W Dataset*

Desde la publicación de dataset en 2019[31] se han publicado diversos artículos que han aplicado distintas técnicas de ML para la clasificación de los eventos anómalos. Marins *et al*[24] utilizan un esquema de cálculo (Figura 2.8) que comparten casi todos los artículos. Cada observación se forma a partir de un ventana de tiempo deslizante (*rolling time window*) de la que se calculan una colección de estadísticos (media, desviación estándar, máximo, etc.). Se aplica una reducción del número de variables (*features*) mediante técnicas como PCA (*Principal Component Analysis*). El dataset ya preprocesado se alimenta a diferentes algoritmos de clasificación. Los mejores resultados se obtienen utilizando Random Forest (RF) consiguiendo precisiones por encima del 90 %.

Brønstad *et al*[7] refinan la metodología anterior utilizando también RF. Turan y Jäschke[30] proponen el uso de un árbol de decisión (*Decision Tree*) que, aunque obtiene resultados ligeramente inferiores a un RF, es mucho más simple e interpretable. Figueirêdo *et al*[11] tienen un planteamiento diferente, ensayan técnicas de aprendizaje no supervisado (*unsupervised*

Figura 2.8: Diagrama de bloques de la aplicación[24]

Learning) para detectar/seleccionar situaciones anómalas que deberán etiquetar los expertos. Carvalho*et al*[9][8] se centran en una de las anomalías, *Flow Instability*), en la selección de variables y en la evaluación de la precisión; alertan del sesgo de similaridad (*similarity bias*) en la selección de los conjuntos de test y entrenamiento. De Salvo *et al*[10] proponen el uso de un gráfico de control de las principales variables (*Control Chart*) para detectar anomalías y utilizan un RF para clasificarlas. Aslam *et al*[1] proponen una metodología donde la novedad se encuentra en el uso de técnicas de *Explainable Artificial Intelligence* (XAI), que permiten interpretar los modelos caja negra (*black box*). Machado *et al*[23] usan un autoencoder *Long Short-Term Memory* (LSTM) y un *one-class Support Vector Machine* (SVM) para clasificar dos de las anomalías (*Spurious DHSV closure* y *Hydrate in Production Line*): la primera con una dinámica rápida (pocos minutos) y la otra lenta (más de una hora); mejoran la precisión y el F1 de estudios anteriores.

Capítulo 3

Materiales y métodos

Todo el proyecto se ha desarrollado en *Jupyter Notebooks* y se puede encontrar en https://github.com/BaltiBoix/3W_TFM. Una *fork* del *github* original del Dataset 3W.

3.1. Descripción del Dataset 3W

En la Figura 3.1 puede verse un esquema simplificado de una instalación típica de extracción de crudo marina.

El dataset 3W[31] está formado por un conjunto de series temporales que describen la operación de un pozo de extracción. Las series tienen como origen:

- Datos reales de pozos. Identifican el pozo con un número.
- Datos generados por simulación.
- Datos manuales generadas por expertos.

Cada serie temporal se almacena en un archivo CSV. Cada observación es una línea en el archivo CSV. La primera línea de cada archivo CSV contiene un encabezado con identificadores de columna. Cada columna de archivos CSV almacena el siguiente tipo de información:

- **timestamp**: Marca de tiempo que actúa como índice
- **P-PDG**: variable de presión en el Medidor Permanente de Fondo de Pozo (PDG) Pa
- **P-TPT**: variable de presión en el Transductor de Temperatura y Presión (TPT) Pa
- **T-TPT**: variable de temperatura en el Transductor de Temperatura y Presión (TPT) ℃
- **P-MON-CKP**: variable de presión aguas arriba del regulador de producción (CKP) Pa

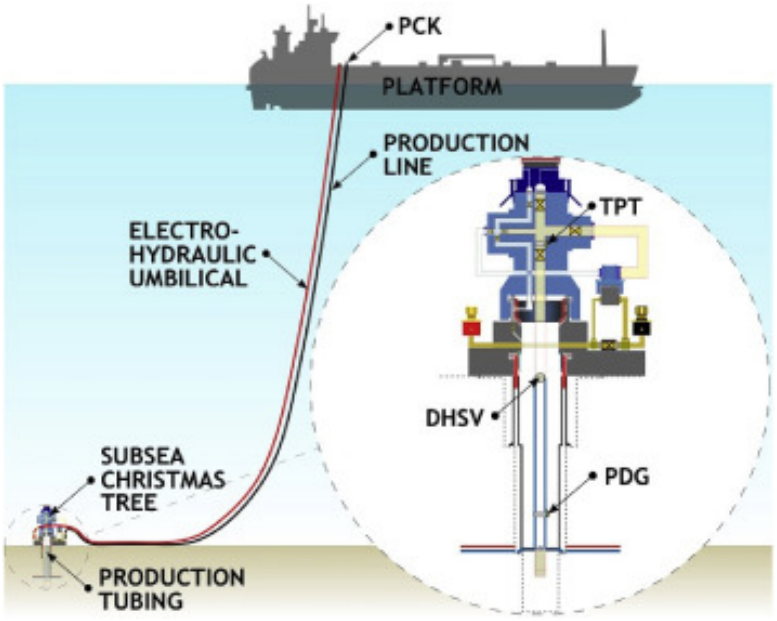


Figura 3.1: Esquema simplificado de un típico pozo marino de flujo natural.[31]

- **T-JUS-CKP**: temperatura variable aguas abajo del regulador de producción (CKP) ℃
- **P-JUS-CKGL**: variable de presión aguas arriba del regulador de gas-lift (CKGL) Pa
- **T-JUS-CKGL**: temperatura variable aguas arriba del regulador de gas-lift (CKGL) ℃
- **QGL**: caudal de gas-lift (inyeccion de gas para incrementar la producción) m^3/s
- **class**: etiquetas de observaciones asociadas con tres tipos de períodos (normal, transitorio del evento anómalo y estado estable defectuoso)

Cada serie está etiquetada a dos niveles por expertos:

- globalmente por la situación normal o anómalo que se desarrolla en la serie.
- Por registro (uno por segundo): normal, transición, anómalo.

En la tabla 3.1 se muestran los eventos anómalos junto con las ventanas de tiempo usadas por los expertos para etiquetarlas. En el apartado 2.2 del artículo de Vargas *et al.*[31] se encuentra una descripción detallada de cada una de ellas.

TYPE OF UNDESIRABLE EVENT	WINDOW SIZE
1 - ABRUPT INCREASE OF BSW	12h
2 – SPURIOUS CLOSURE OF DHSV	5min–20min
3 – SEVERE SLUGGING	5h
4 – FLOW INSTABILITY	15min
5 – RAPID PRODUCTIVITY LOSS	12h
6 – QUICK RESTRICTION IN PCK	15min
7 – SCALING IN PCK	72h
8 – HYDRATE IN PRODUCTION LINE	30min–5h

Tabla 3.1: *Tamaño estimado de las ventanas de tiempo utilizadas para confirmar la ocurrencia de eventos no deseados[31].*

3.2. Análisis exploratorio de las series reales

En la figura 3.2 se muestra de forma gráfica dos series correspondientes al pozo 1 en situación normal y en un episodio de inestabilidad de flujo. Las variables en la situación normal muestran un valor estable afectado por lo que se podría denominar como ruido de fondo mientras que en la situación de inestabilidad de flujo el valor oscila. Se tendrá que utilizar una ventana de tiempo deslizante para distinguirlos.

En la figura 3.3 se muestra la distribución de series reales(R), simuladas(S) y manuales(D) por tipo de evento. Hay que resaltar que con datos reales solo están bien representadas la situación normal (0) y la inestabilidad de flujo (4). Para otras situaciones anómalas se necesita utilizar datos simulados. En la figura 3.4 se muestran los datos de una serie simulada y una real para la situación de Rápida Pérdida de Producción (5). Se observa que los datos simulados no incorporan el ruido de fondo típico de los reales.

En la figura 3.5 se muestra el cronograma de adquisición de los datos reales por tipo de evento y pozo. Se observan dos etapas centradas en 2014 y 2017.

En la tabla 3.2 se muestra el porcentaje de valores no disponibles en las series temporales reales del dataset. Las variables *P-PDG*, *P-TPT* y *T-TPT* están prácticamente siempre disponibles.

label	rcount	P-PDG	P-TPT	T-TPT	P-MON-CKP	T-JUS-CKP	P-JUS-CKGL	T-JUS-CKGL	QGL	class
0	9956791	0.04	0.04	0.04	10.13	14.79	25.63	100.00	25.48	0.00
1	118294	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.86
2	158680	0.39	0.39	0.39	69.85	86.66	90.01	100.00	86.66	0.65
3	569152	0.10	0.10	0.10	0.12	0.10	0.12	100.00	0.10	0.00
4	2462076	0.03	0.04	0.04	0.03	0.05	50.51	100.00	23.54	0.00
5	361998	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.40
6	54212	0.00	0.00	0.00	0.00	0.00	11.70	100.00	11.70	1.15
7	271708	0.13	0.13	0.13	0.13	0.13	0.13	100.00	0.13	0.23

Tabla 3.2: Porcentaje de Nan en el Dataset 3W

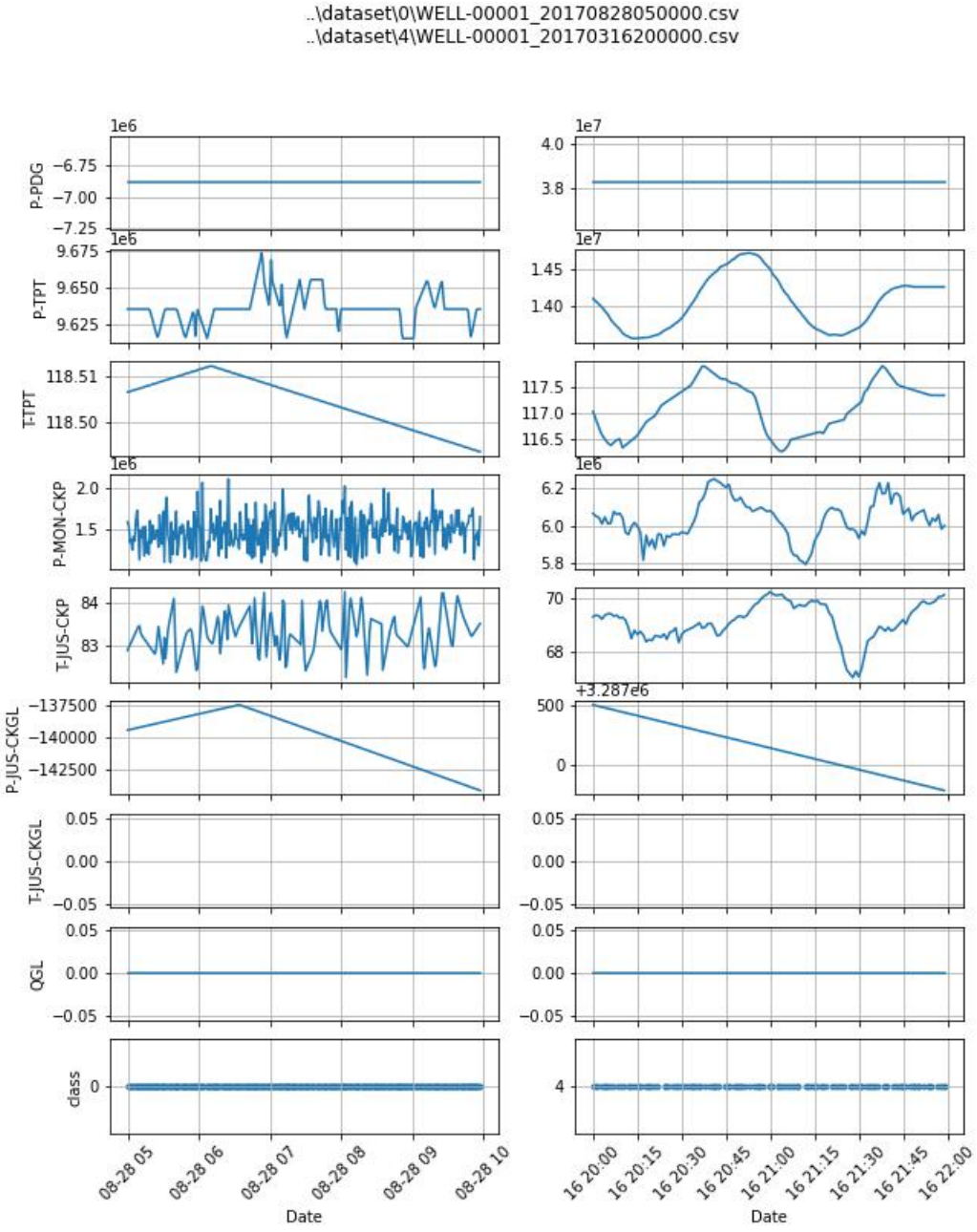


Figura 3.2: Ejemplo de series temporales del dataset 3W

SOURCE INSTANCE LABEL	REAL	SIMULATED	HAND-DRAWN	TOTAL
0 - Normal Operation	597	0	0	597
1 - Abrupt Increase of BSW	5	114	10	129
2 - Spurious Closure of DHSV	22	16	0	38
3 - Severe Slugging	32	74	0	106
4 - Flow Instability	344	0	0	344
5 - Rapid Productivity Loss	12	439	0	451
6 - Quick Restriction in PCK	6	215	0	221
7 - Scaling in PCK	4	0	10	14
8 - Hydrate in Production Line	0	81	0	81
TOTAL	1022	939	20	1981

Figura 3.3: Distribución de las series temporales del dataset 3W

3.3. Preprocesamiento de los datos

Las series temporales del Dataset tienen una frecuencia de un registro por segundo. Ya que los eventos que se pretende detectar requieren de unas ventanas de tiempo de entre 5min y 72h [31] no parece que la dinámica del proceso requiera de una frecuencia de muestreo tan alta. Se ha decidido reducir el tamaño del dataset utilizando el promedio minutal de las variables y para la etiqueta **class** el valor más frecuente en el minuto.

Las series temporales tienen como origen datos reales de 18 pozos diferentes, series simuladas y otras rellenadas manualmente. Para poder utilizar el dataset en conjunto las variables se han normalizado (*StandardScaler*). Los datos no disponibles (*Nan*) se han estimado con el método *forward fill* cuando en una serie hay menos del 20 %; si hay más, toda la serie se sustituye por cero. No se ha aplicado ninguna técnica de eliminación de *outliers* al considerar que interferiría con el objetivo de detectar eventos anómalos y drift. En el caso de datos puntuales extraños el promedio de los datos por segundo a minuto los absorberá en parte. En el análisis visual, aunque no ha sido completo, no se han detectado.

Para gestionar los ficheros de las series temporales se ha programado la *python class* **d3w** que permite seleccionar las series según su origen, evento o pozo y generar conjuntos para entrenamiento, test y validación.

Las series seleccionadas con **d3w** se alimenta a una *python class* que genera los datos de forma continua concatenando los registros de cada serie temporal para cada tipo de modelo.

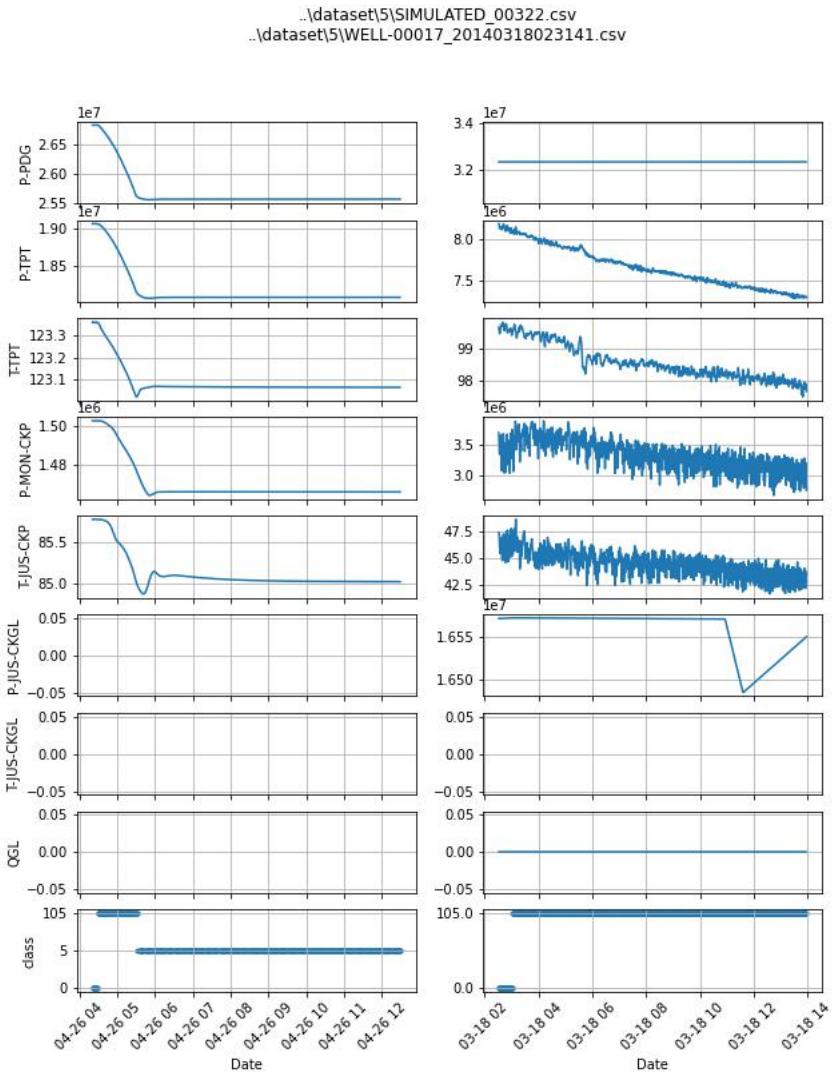


Figura 3.4: Ejemplo de serie temporal real y simulada

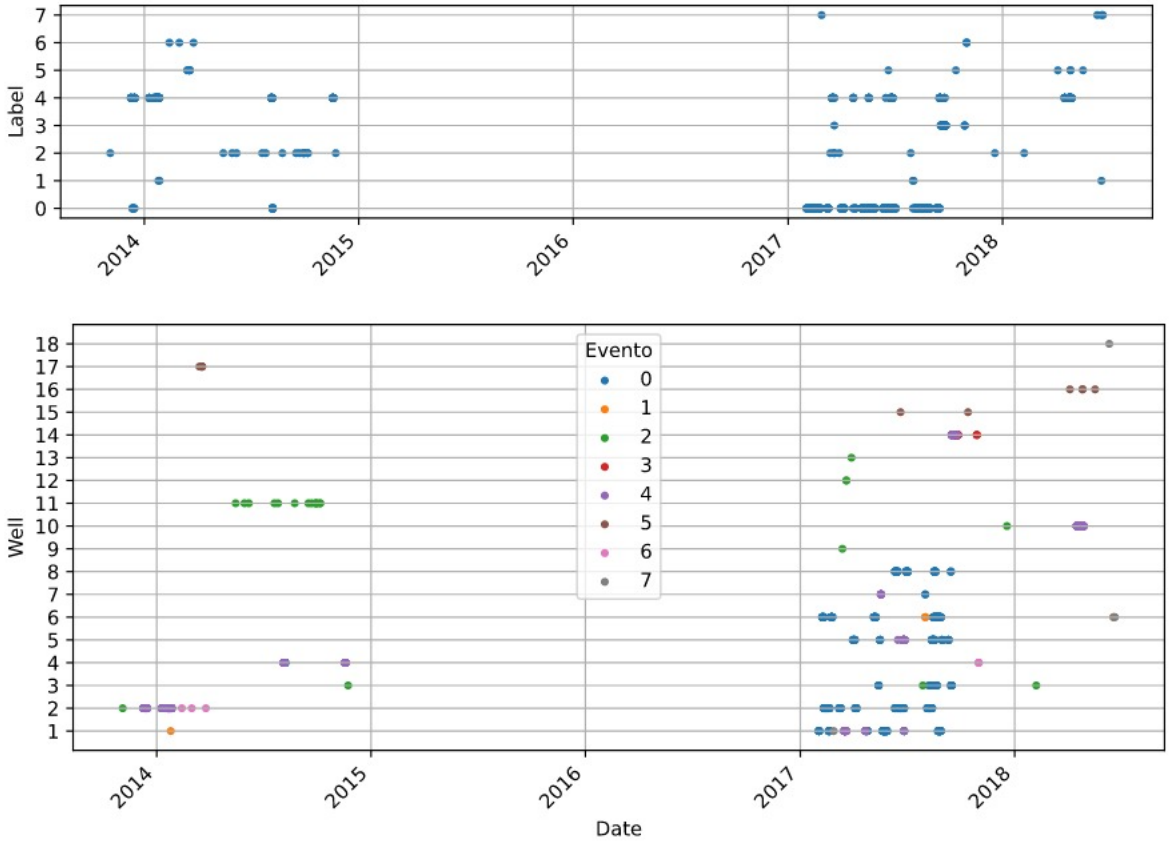


Figura 3.5: Cronograma de la adquisición de datos reales del dataset 3W

3.4. Métodos Online

En esta sección se utiliza el dataset 3W para simular un entorno en tiempo real en el que los datos se procesan registro a registro. Es un ejercicio teórico en el que se hacen algunas aproximaciones que nos alejan del problema real:

- Se considera que se dispone de la etiqueta del registro con solo un minuto de retraso mientras que en el caso real el etiquetado lo han realizado expertos a posteriori.
- Se concatenan las series temporales como si fuera una sola cuando pueden distar entre ellas años (ver el cronograma en la figura 3.5).
- Las series que tienen como origen una simulación o están creadas manualmente no las podemos situar en el tiempo.

3.4.1. Modelo base

Se ha hecho un uso extensivo del módulo River [25]. Para el preprocesamiento de los registros se han utilizado las rutinas que normalizan y calculan variables estadísticas incrementalmente y que, por tanto, no usan información del futuro para el calculo actual. Se utiliza la información de una ventana de tiempo deslizante (*rolling time window*) pero en vez de usar los datos directamente se simplifican en la media, desviación estándar, máximo y mínimo. El uso de datos instantáneos (cada segundo) sería lo que se asemejaría más a recibir los datos directamente de un SCADA (*Supervisory Control and Data Acquisition*) pero resulta extremadamente lento y se ha optado por usar medias por minuto calculadas con el módulo Pandas. Cada vez que cambia la serie temporal se reinician las variables y no se utilizan en el aprendizaje durante un periodo equivalente a una ventana de tiempo. Las variables preprocesadas se alimentan a un modelo que predice una etiqueta que podemos comparar con la etiqueta que determina el experto y, paso seguido, utilizar para actualizar el modelo según se muestra en el esquema 3.6.

Al evaluar los eventos anómalos uno a uno se utilizan las series etiquetadas con ese evento junto con las correspondientes a una operación normal. El dataset resultante está desbalanceado a favor de esta última. Para corregirlo se utiliza la rutina de River imblearn RandomOverSampler como *wrapper* para clasificadores.Entrenará al clasificador sobremuestreando el flujo de observaciones para que la distribución de clases vista siga una distribución deseada dada. La implementación es una versión discreta del muestreo de rechazo inverso. El modelo escogido ha sido el clasificador multiclase tree.HoeffdingTreeClassifier:

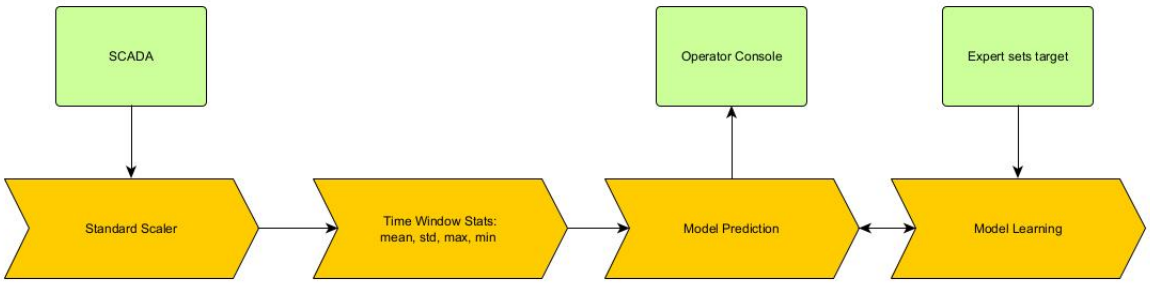


Figura 3.6: Diagrama del modelo online utilizado

- Un *Hoeffding Tree* es un algoritmo de inducción de árbol de decisión incremental que es capaz de aprender de flujos de datos masivos, suponiendo que los ejemplos de generación de distribución no cambien con el tiempo. Los *Hoeffding Tree* aprovechan el hecho de que una pequeña muestra a menudo puede ser suficiente para elegir un atributo de división óptimo. Esta idea se apoya matemáticamente en el límite de Hoeffding, que cuantifica el número de observaciones (en nuestro caso, ejemplos) necesarias para estimar algunas variables estadísticas dentro de una precisión dada (en nuestro caso, la bondad de un atributo). Una característica teóricamente atractiva de *Hoeffding Tree* que no comparten otros árboles de decisiones incrementales es que tiene un rendimiento garantizado. Usando el límite de Hoeffding, se puede mostrar que su salida es asintóticamente casi idéntica a la de un clasificador no incremental usando infinitos ejemplos. La implementación en River[25] está basada en MOA[4][18].

- Aunque la ejecución del modelo se ha realizado para cada evento anómalo independientemente se requiere un clasificador multiclase ya que algunos distinguen tres situaciones (0: Operación Normal, 10x: transición, x: Evento anómalo).

River permite usar modelos de clasificación binarios en entornos multiclase con el uso de los *wrappers* multiclass.OneVsOneClassifier y multiclass.OneVsRestClassifier. Se han probado los modelos *linear_model.LogisticRegression* y *linear_model.ALMAClassifier*. Se ha comparado la precisión obtenida para cada modelo ajustado en modo sólo predicción. Se comportan peor que el Hoeffding Tree.

3.4.2. Detección de *drift*

El siguiente paso en este trabajo ha sido introducir un detector de *drift* en el esquema 3.6 de procesamiento del dataset visto anteriormente. Cuando se detecta *drift* se reinicia el aprendizaje

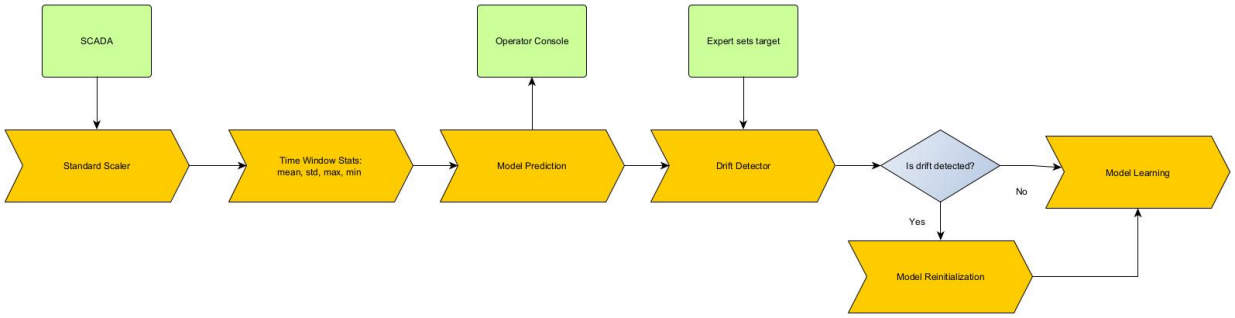


Figura 3.7: Diagrama del modelo online utilizado con detección de *drift*

.

del modelo siempre que el modelo haya visto un mínimo de registros. Se busca observar si se obtiene una mejora de precisión global. El esquema modificado se muestra en la figura 3.7.

El detector de *drift* recibe la etiqueta predicha y la real del experto y estima si se ha producido un cambio en la distribución del error. El módulo River cuenta con varios métodos de detección que se han evaluado con el dataset 3W:

- **ADWIN** (ADaptive WINdowing) es un método popular de detección de *drift*. ADWIN mantiene una ventana de longitud variable de elementos recientes. Esta ventana se divide a su vez en dos subventanas (W_0, W_1) que se utilizan para determinar si se ha producido un cambio en la distribución de los datos. ADWIN compara el promedio de W_0 y W_1 para confirmar que corresponden a la misma distribución. El *concept drift* se detecta si la igualdad de distribución ya no se mantiene. Al detectar un desvío, W_0 se reemplaza por W_1 y se inicializa uno nuevo W_1. ADWIN utiliza un valor significativo $\delta \in (0, 1)$ para determinar si las dos subventanas corresponden a la misma distribución [3].

- **DDM** (Método de detección de deriva) es un método de detección de *concept drift* basado en la premisa del modelo de aprendizaje PAC, que la tasa de error del clasificador disminuirá a medida que aumente la cantidad de muestras analizadas, siempre que la distribución de datos sea estacionaria.

  Si el algoritmo detecta un aumento en la tasa de error, que supera un umbral calculado, se detecta un cambio o el algoritmo advertirá al usuario que puede ocurrir un cambio en un futuro próximo, lo que se denomina zona de advertencia (*Warning Zone*).

  El umbral de detección se calcula en función de dos estadísticas, obtenidas cuando es $(p_i + s_i)$ mínimo:

  - p_{min}: La tasa de error mínima registrada.
  - s_{min}: La desviación estándar mínima registrada.

En el instante i, el algoritmo de detección utiliza:

- p_i: La tasa de error en el instante i.
- s_i: La desviación estándar en el instante i.

Las condiciones para ingresar a la zona de advertencia y detectar cambios son las siguiente:

- si $p_i + s_i \geq p_{min} + w_l * s_{min} \Rightarrow$ Zona de advertencia
- si $p_i + s_i \geq p_{min} + d_l * s_{min} \Rightarrow$ Cambio detectado

En las expresiones anteriores, w_l y d_l representan, respectivamente, los umbrales de advertencia y deriva (*drift*)[13].

- **EDDM** (*Early Drift Detection Method*) tiene como objetivo mejorar la tasa de detección del *concept drift* gradual en DDM, manteniendo un buen rendimiento en la abrupta.

  Este método funciona al realizar un seguimiento de la distancia promedio entre dos errores en lugar de solo la tasa de error. Para esto, es necesario realizar un seguimiento de la distancia promedio y la desviación estándar, así como la distancia máxima y la desviación estándar máxima.

  El algoritmo funciona de manera similar al algoritmo DDM, al realizar un seguimiento de las estadísticas únicamente. Funciona con la distancia media ($p_i^{'}$) y la desviación estándar ($s_i^{'}$), así como $p_{max}^{'}$ y $s_{max}^{'}$, que son los valores de $p_i^{'}$ y $s_i^{'}$ cuándo ($p_i^{'} + 2 * s_i^{'}$) alcanza su máximo.

  Al igual que DDM, hay dos valores de umbral que definen el límite entre no cambio, zona de advertencia y desviación detectada:

  - si $(p_i^{'} + 2 * s_i^{'})/(p_{max}^{'} + 2 * s_{max}^{'}) < \alpha \Rightarrow$ Zona de advertencia
  - si $(p_i^{'} + 2 * s_i^{'})/(p_{max}^{'} + 2 * s_{max}^{'}) < \beta \Rightarrow$ Cambio detectado

  α y β por defecto valen 0.95 y 0.9, respectivamente[2].

- **HDDM_A** es un método de detección de *drift* basado en la desigualdad de Hoeffding que utiliza el promedio de entrada como estimador[4][12].

- **HDDM_W** es un método de detección de *drift* en línea basado en los límites de McDiarmid. Utiliza la media móvil ponderada exponencialmente (EWMA) como estimador [4][12].

Capítulo 4

Experimentación y Resultados

En la figura 4.1 se muestra una representación gráfica de los primeros 4 de 12 niveles del *Hoeffding Tree* resultante del procesamiento de las series de datos reales con etiquetas 0 (Operación Normal) y 4 (Flujo inestable).

En las figuras 4.4 y 4.7 se puede observar la precisión obtenida en el proceso de aprendizaje incremental y en la evaluación del mismo dataset usando el modelo ya entrenado para el Hoeffding Tree y el linear_model.ALMAClassifier.

Los episodios de funcionamiento anómalo del proceso de extracción de crudo, cuando se producen, no son puntuales sino que tienen una evolución que se alarga en el tiempo. Este comportamiento tiene como consecuencia que las etiquetas muestren una alta autocorrelación que, por otro lado, es muy frecuente en las series temporales. En la figura 4.8 se compara la precisión del modelo con respecto al modelo de River *dummy.NoChangeClassifier* que simplemente asigna la etiqueta anterior al registro actual. En la parte superior se muestra el cronograma de las series temporales reales utilizadas para el evento anómalo 4 (*Flow Instability*). Repitiendo el ejercicio para el evento anómalo 5 (*Rapid Productivity Loss*) en el que la mayor parte de las

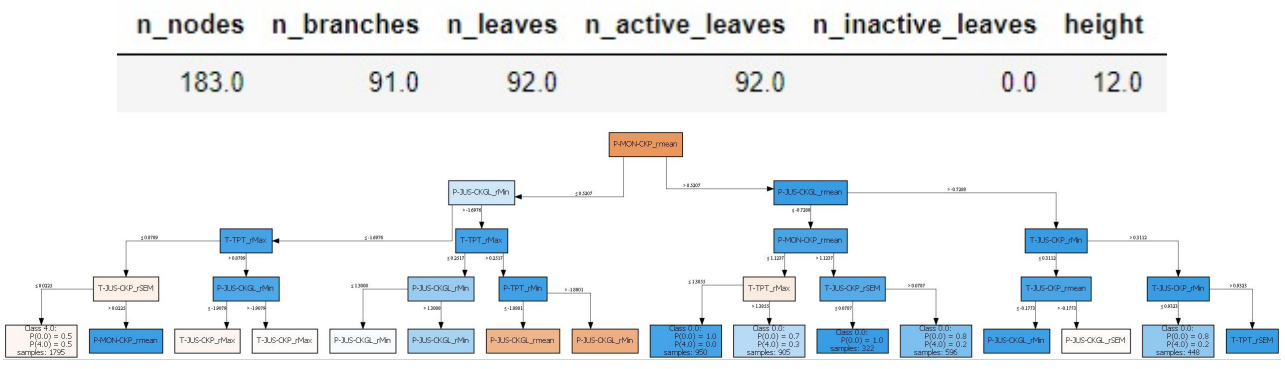


Figura 4.1: Modelo Hoeffding Tree para la detección de *Flow Instability* con datos reales

BalancedAccuracy: 93.75%

	Precision	Recall	F1	Support
0.0	98.35%	95.07%	96.68%	149201
4.0	79.82%	92.43%	85.66%	31501
Macro	89.08%	93.75%	91.17%	
Micro	94.61%	94.61%	94.61%	
Weighted	95.12%	94.61%	94.76%	

94.61% accuracy

Figura 4.2: HT Train Accuracy

BalancedAccuracy: 88.17%

	Precision	Recall	F1	Support
0.0	96.45%	92.44%	94.40%	149201
4.0	70.07%	83.90%	76.37%	31501
Macro	83.26%	88.17%	85.38%	
Micro	90.95%	90.95%	90.95%	
Weighted	91.85%	90.95%	91.26%	

Figura 4.3: HT Evaluation Accuracy

Figura 4.4: Precisión del Hoeffding Tree

BalancedAccuracy: 99.36%

	Precision	Recall	F1	Support
0.0	99.79%	99.68%	99.74%	149201
4.0	98.50%	99.03%	98.76%	31501
Macro	99.15%	99.36%	99.25%	
Micro	99.57%	99.57%	99.57%	
Weighted	99.57%	99.57%	99.57%	

Figura 4.5: ALMA Train Accuracy

BalancedAccuracy: 59.34%

	Precision	Recall	F1	Support
0.0	90.61%	36.70%	52.24%	149201
4.0	21.47%	81.98%	34.03%	31501
Macro	56.04%	59.34%	43.14%	
Micro	44.60%	44.60%	44.60%	
Weighted	78.56%	44.60%	49.07%	

Figura 4.6: ALMA Evaluation Accuracy

Figura 4.7: Precisión del linear_model.ALMAClassifier

series son simuladas y se han ordenado aleatoriamente, se observa el mismo fenómeno porque la autocorrelación se da dentro de cada serie (Figura 4.9.

En los gráficos de la figura 4.10 se puede ver la aplicación de los métodos de detección de *drift* descritos anteriormente en sección 3.4.2 a las series de datos reales con etiquetas 0 (Operación Normal) y 4 (Flujo inestable) ordenados por fecha. Se reinicializa el aprendizaje del modelo cuando se detecta *drift* y el modelo al menos ha visto 5000 registros. Se observa que la mayor parte de las detecciones ocurren en aproximadamente los mismos puntos y en todos los casos mejora la precisión.

Se ha comprobado que todos los métodos, ajustando sus parámetros de configuración, detectan *drift* en aproximadamente los mismos puntos. ADWIN es más fácil de ajustar que los demás; quizás porque es más configurable.

El comportamiento del resto de eventos anómalos no es significativo porque como se muestra en la figura 3.3 se dispone de pocas series reales. se muestran los resultados para los eventos 1 a 3 (ver tabla 3.1) con el método de detección ADWIN en la figura 4.11.

Repitiendo el ejercicio para el evento anómalo 5 (*Rapid Productivity Loss* con series simuladas y ordenadas las series aleatoriamente (no los registros) con los mismos ajustes de ADWIN que se han utilizado anteriormente se obtiene un resultado absurdo (Figura 4.12.

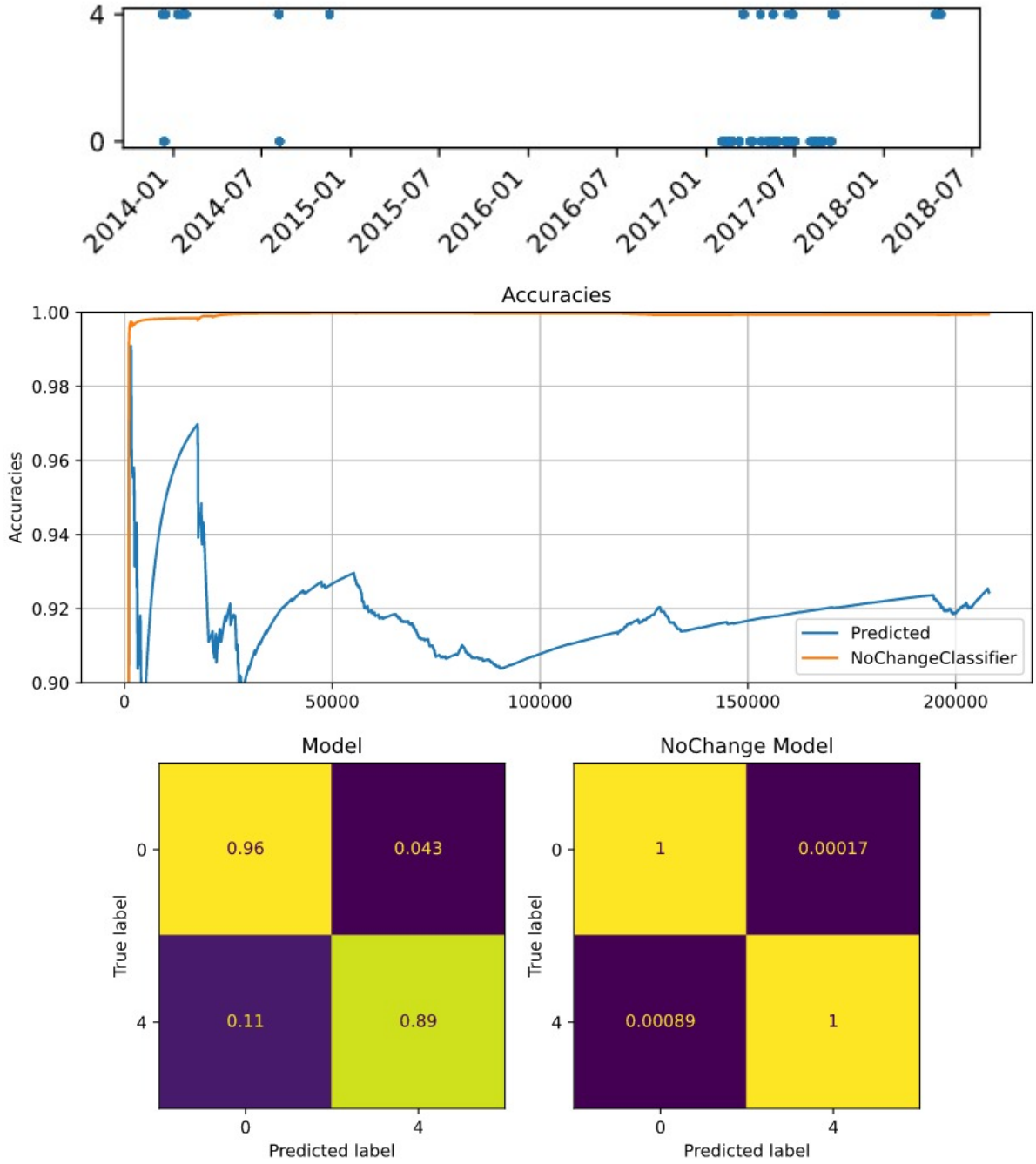


Figura 4.8: Comparación del modelo Hoeffding Tree con el NoChange para *Flow Instability* con datos reales

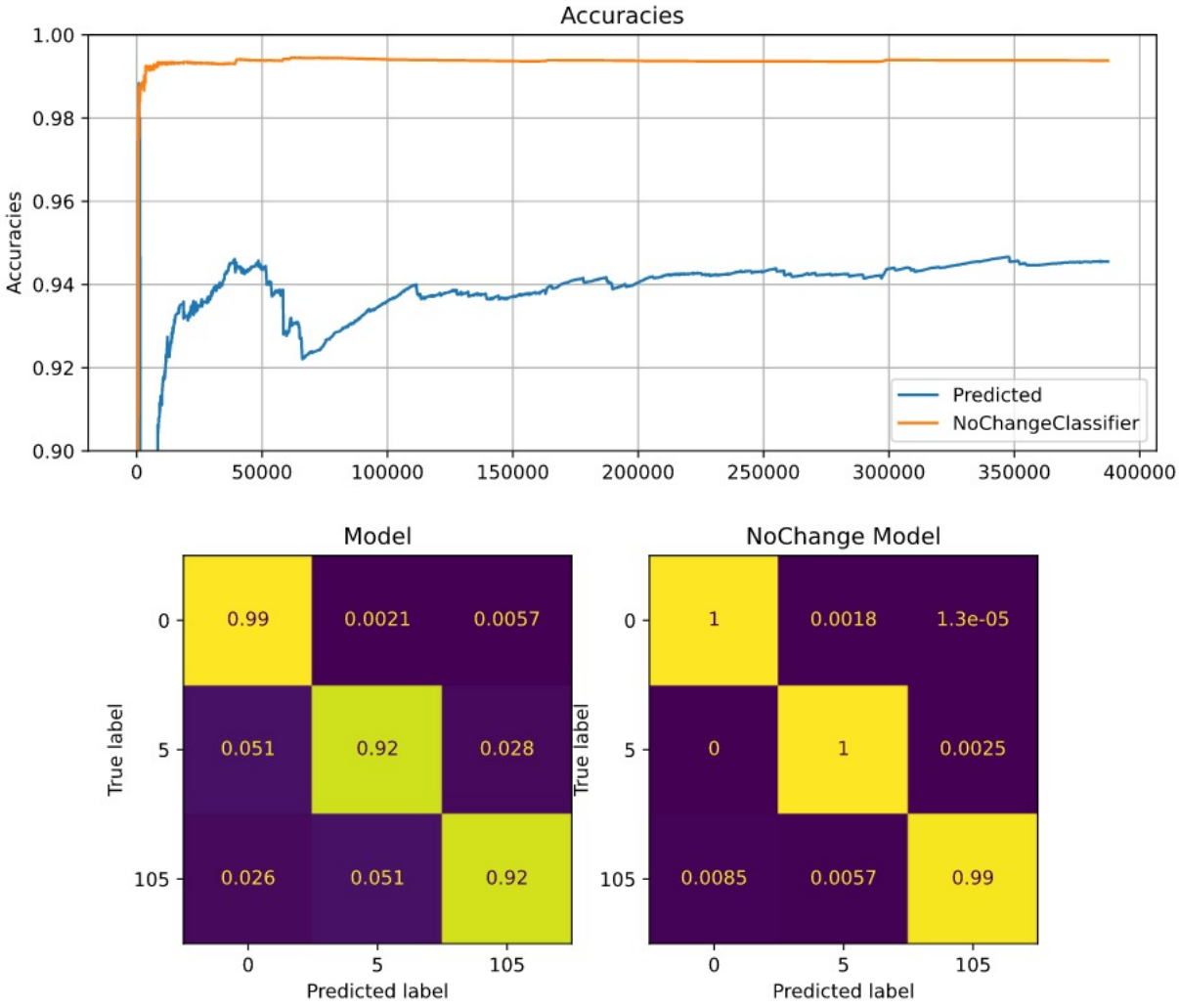
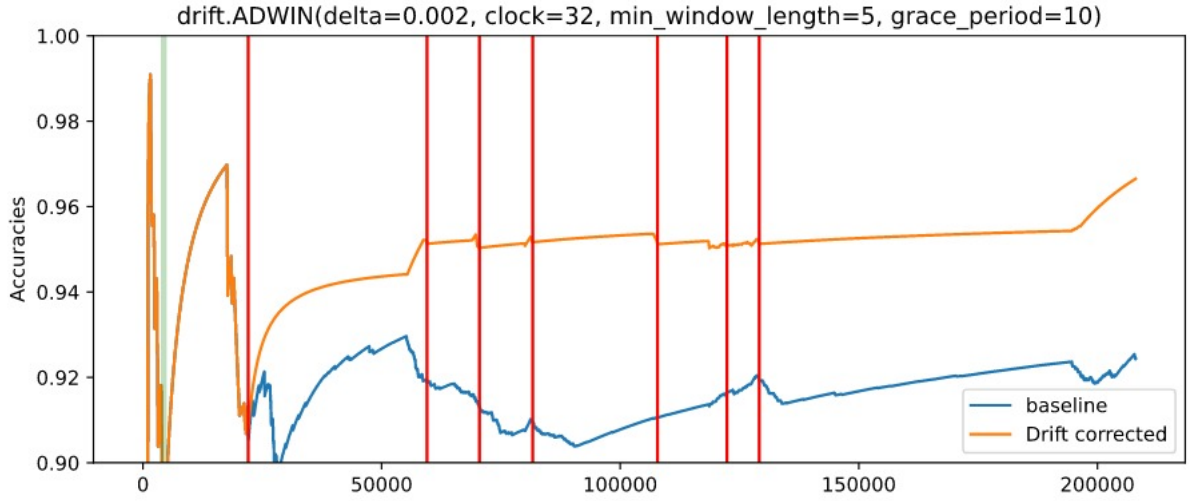
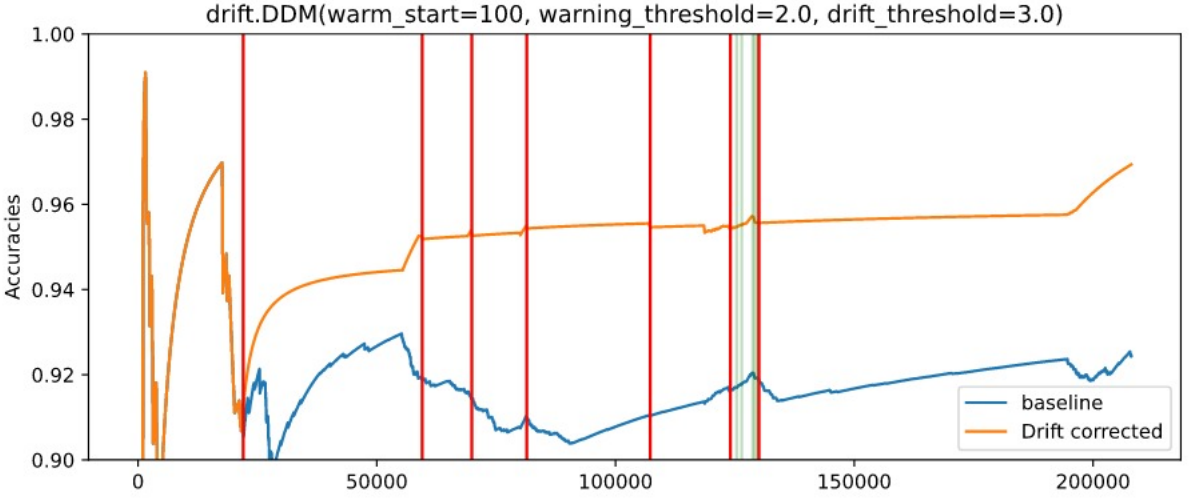
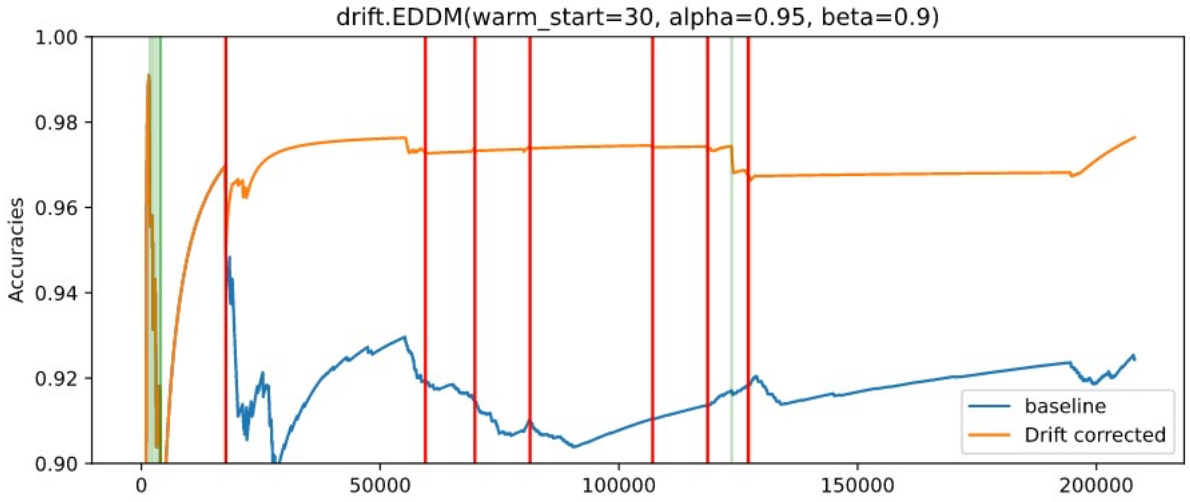


Figura 4.9: Comparación del modelo Hoeffding Tree con el NoChange para *Rapid Productivity Loss* con datos reales y simulados ordenados aleatoriamente


drift.ADWIN(delta=0.002, clock=32, min_window_length=5, grace_period=10)
Accuracies
1.00
0.98
0.96
0.94
0.92
0.90
0
50000
100000
150000
200000
baseline
Drift corrected


drift.DDM(warm_start=100, warning_threshold=2.0, drift_threshold=3.0)
Accuracies
1.00
0.98
0.96
0.94
0.92
0.90
0
50000
100000
150000
200000
baseline
Drift corrected


drift.EDDM(warm_start=30, alpha=0.95, beta=0.9)
Accuracies
1.00
0.98
0.96
0.94
0.92
0.90
0
50000
100000
150000
200000
baseline
Drift corrected

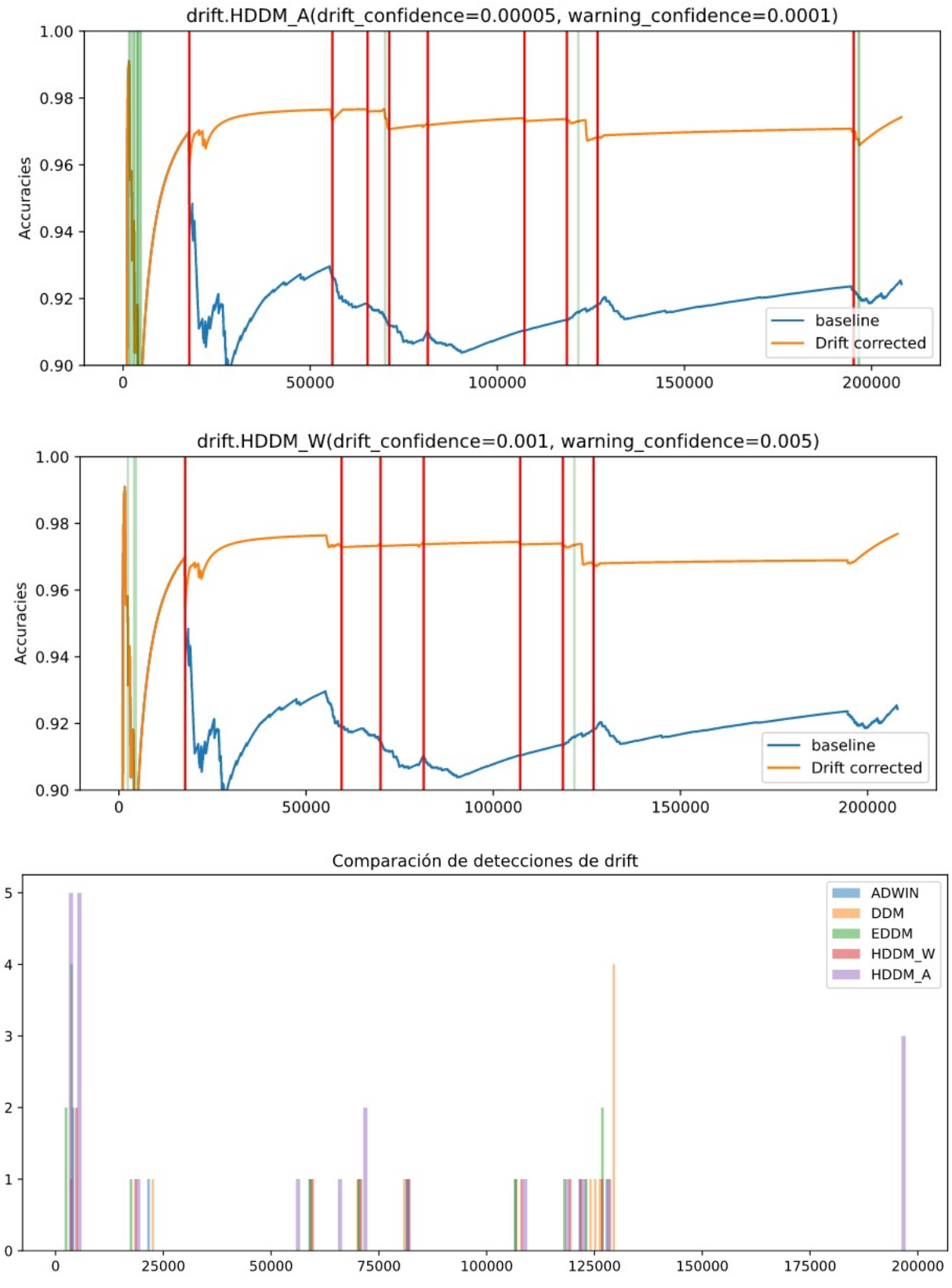


Figura 4.10: Detección de *drift* con diferentes métodos para *flow instability*

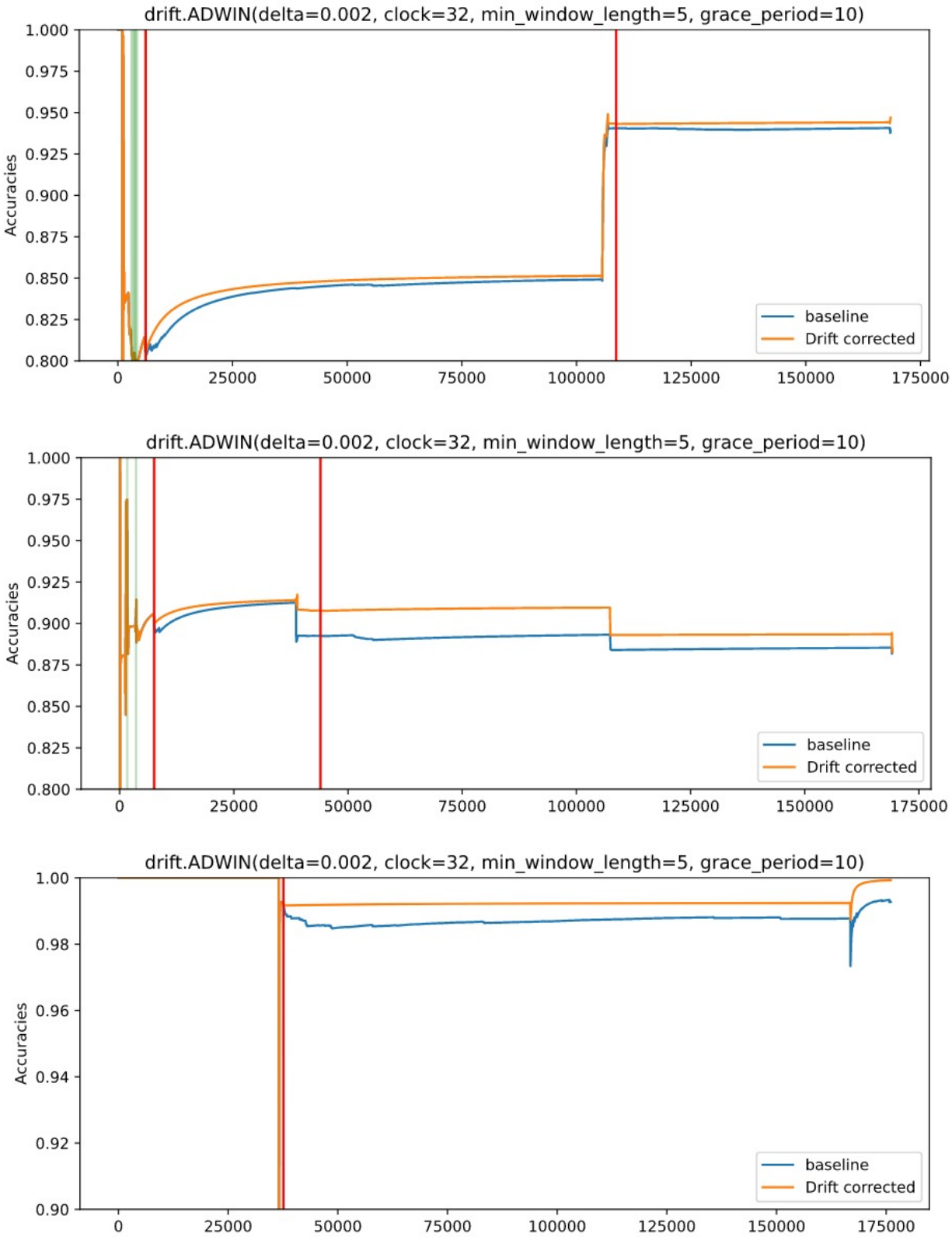


Figura 4.11: Detección de *drift* con ADWIN para los eventos anómalos 1 (Abrupt Increase of BSW), 2 (Spurious Closure OF DHSV) y 3 (Severe Slugging)

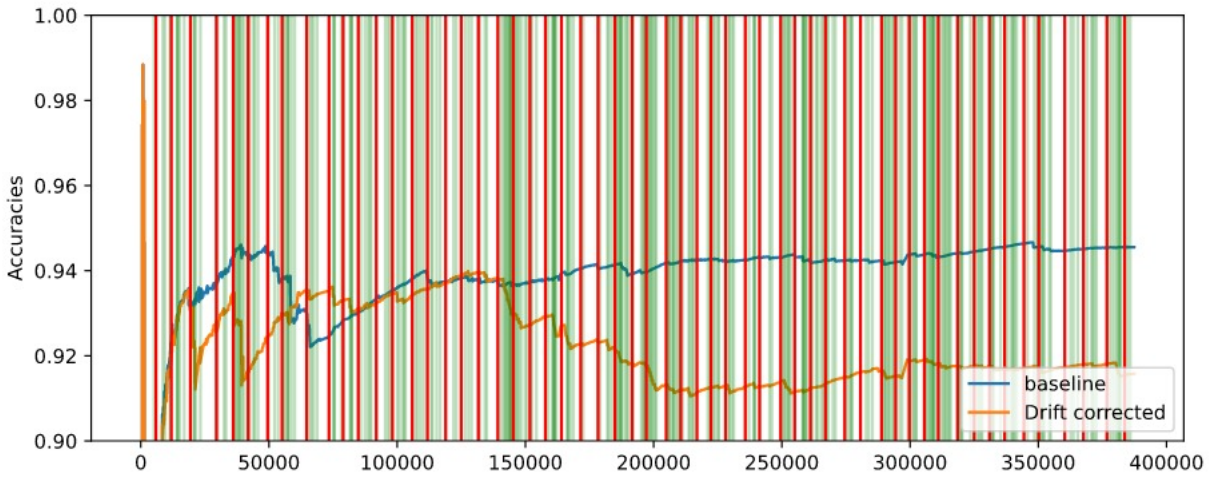


Figura 4.12: Detección de *drift* con ADWIN para el evento anómalo 5 (*Rapid Productivity Loss*) con series simuladas

Capítulo 5

Conclusiones del TFM

De los estudios publicados hasta la fecha podemos concluir que es posible desarrollar una aplicación de soporte a la operación de los pozos de petróleo *offshore* que explote los modelos offline ajustados con el dataset 3W (En el apéndice A se puede observar que utilizando *Random Forest* y redes neuronal RNN y CNN relativamente sencillas se obtienen buenos resultados).

El enfoque novedoso de este TFM es la aplicación de técnicas de aprendizaje incremental y de detección de *drift*.

A partir de las series temporales del dataset 3W se ha emulado la recepción de los datos desde un SCADA (Supervisory Control and Data Acquisition) y su procesamiento utilizando las rutinas de *stream learning* del módulo River[25] de preprocesamiento y aprendizaje incremental (Figura 3.6).

Al proceso anterior se le ha implementado un detector de *concept drift* que reinicializa el aprendizaje (Figura 3.7). Se observa una clara mejora cuando se utilizan datos reales de dataset ordenados por fecha. Lamentablemente únicamente en el caso del evento anómalo *Flow Instabilility* se dispone de suficientes datos reales (no simulados o manuales).

Una posible explicación del *concept drift* detectado podría ser que los expertos que han etiquetado las series hayan cambiado con los años o haya cambiado el criterio aplicado.

Aunque los métodos de detección de *drift* han dado resultados similares, ADWIN es el más configurable y permite un ajuste más fino.

De los resultados obtenidos se puede concluir que es posible realizar una detección en tiempo real fiable de eventos anómalos durante el proceso de extracción de crudo que era el objetivo principal del TFM. En cuanto al objetivo secundario de considerar de forma adecuada la presencia del fenómeno *concept drift* durante el proceso de aprendizaje, podemos afirmar que se consigue detectar aunque no tenemos evidencia de si es real o de los motivos que lo generan.

Hay que resaltar que el uso de un proceso de aprendizaje incremental como el que se ha emulado sólo sería posible realmente si dispusiéramos de las etiquetas con un minuto de retraso.

En realidad el etiquetado lo realiza un experto offline. En cualquier caso sería válido para modelos cuyo objetivo sea hacer predicciones (*forecasting*) ya que se evita el etiquetado offline.

5.1. Lineas de Trabajo Futuras

Ideas para trabajos futuros:

- Ajustar un modelo offline con datos de los años 2014 y 2015 con un detector de *drift* aplicado sobre los datos de entrada al modelo (en vez de sobre el error en la predicción)[28] que reportara si la predicción deja de ser fiable con datos de los años posteriores.
- El dataset 3W se actualiza periódicamente y, por tanto, se podrían actualizar los modelos o comprobar si siguen siendo válidos los modelos actuales.
- Crear una aplicación que detecte *concept drift* en los datos del SCADA del proceso de extracción de crudo y aísle series temporales que posteriormente un experto pueda etiquetar. Machado *et al*[23] usan un autoencoder *Long Short-Term Memory* (LSTM) que detecta un evento anómalo en función del error entre la entrada y salida del autoencoder.

Apéndice A

Métodos Offline

Para saber de antemano el rendimiento de un modelo tradicional sobre los datos, y para que sirva de referencia, se han ajustado diferentes modelos offline al dataset pese a que no es un objetivo del TFM. A continuación se presenta un breve resumen.

A.1. Redes Neuronales

Se han ensayado tres modelos de redes neuronales en el entorno Keras relativamente sencillos que se muestran en la figura A.1. Se ha optado por modelos multiclase utilizando las series temporales de todos los orígenes (reales, simuladas y manuales).

Model: "dnn_model"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(64, 480)	0
layer_normalization (LayerNormalization)	(64, 480)	960
dense (Dense)	(64, 256)	123136
dropout (Dropout)	(64, 256)	0
dense_1 (Dense)	(64, 128)	32896
dropout_1 (Dropout)	(64, 128)	0
dense_2 (Dense)	(64, 64)	8256
dropout_2 (Dropout)	(64, 64)	0
dense_3 (Dense)	(64, 17)	1105
softmax (Softmax)	(64, 17)	0

Total params: 166,353
Trainable params: 166,353
Non-trainable params: 0

Model: "cnn_model"

Layer (type)	Output Shape	Param #
layer_normalization (LayerNormalization)	(64, 30, 16)	32
conv1 (Conv1D)	(64, 30, 128)	4224
MP1 (MaxPooling1D)	(64, 15, 128)	0
drop1 (Dropout)	(64, 15, 128)	0
conv2 (Conv1D)	(64, 15, 128)	32896
MP2 (MaxPooling1D)	(64, 7, 128)	0
drop2 (Dropout)	(64, 7, 128)	0
conv3 (Conv1D)	(64, 7, 128)	32896
MP3 (MaxPooling1D)	(64, 3, 128)	0
drop3 (Dropout)	(64, 3, 128)	0
flatten (Flatten)	(64, 384)	0
dense (Dense)	(64, 17)	6545
softmax (Softmax)	(64, 17)	0

Total params: 76,593
Trainable params: 76,593
Non-trainable params: 0

Model: "rnn_model"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(64, 128)	74240
dense1 (Dense)	(64, 17)	2193
softmax (Softmax)	(64, 17)	0

Total params: 76,433
Trainable params: 76,433
Non-trainable params: 0

Figura A.1: Redes neuronales ensayadas

Se ha programado la *python class* **kmodel** que permite definir externamente un modelo keras y ensayarlo con el dataset. Se ha utilizado un tamaño de batch de 64 y una ventana de tiempo de 30 minutos. El *learning rate* se ha adaptado a cada tipo de modelo. No se ha hecho una optimización formal de los hiperparámetros por los largos tiempos de ejecución.

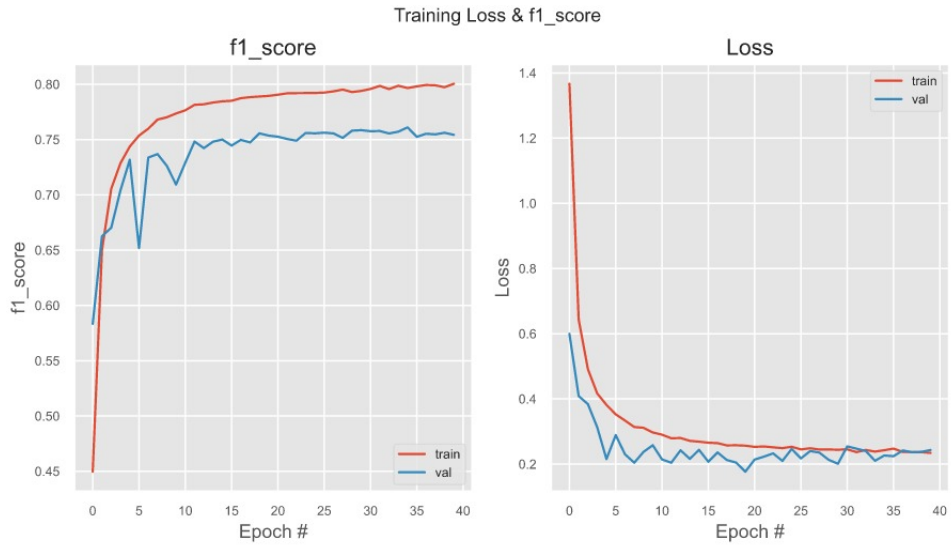


Figura A.2: Evolución del aprendizaje de la RNN

Los mejores resultados se obtienen con la RNN y la CNN.

En la figura A.2 podemos ver la evolución del aprendizaje para la RNN y en la figura A.3 los resultados para el conjunto de test.

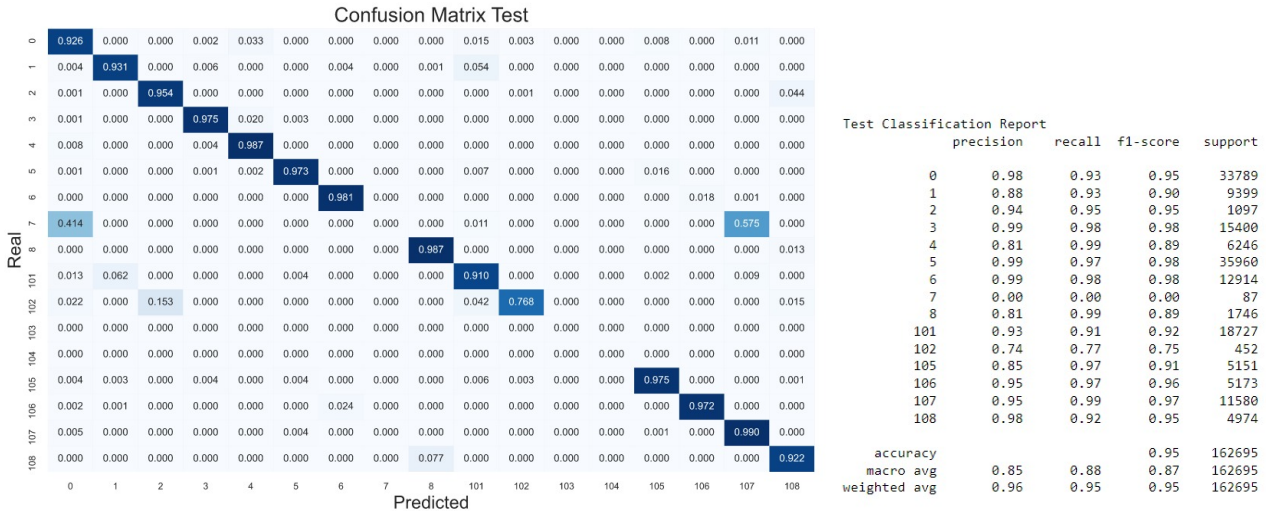


Figura A.3: Resultados de la RNN para el conjunto de test

A.2. Modelos de Sklearn de aprendizaje incremental

Se han ensayado modelos de Sklearn que permiten el aprendizaje incremental. Eso es, que no requieren cargar todos los datos de una vez y evitan saturar la memoria. Se han probado

	precision	recall	f1-score	support
0	0.81	0.92	0.87	33789
1	0.87	0.84	0.86	9399
2	0.99	0.95	0.97	1097
3	0.97	0.91	0.94	15400
4	0.67	0.77	0.72	6246
5	1.00	0.97	0.98	35960
6	0.99	0.99	0.99	12914
7	1.00	0.83	0.91	87
8	0.84	0.96	0.90	1746
101	0.88	0.86	0.87	18727
102	0.99	0.56	0.71	452
105	0.91	0.97	0.94	5151
106	0.97	0.96	0.97	5173
107	0.95	0.76	0.84	11580
108	0.98	0.87	0.92	4974
accuracy			0.91	162695
macro avg	0.92	0.87	0.89	162695
weighted avg	0.91	0.91	0.91	162695

Figura A.4: Resultado de aplicar ipca+RF al conjunto de test

los modelos *SGDClassifier* y *PassiveAggressiveClassifier* con resultados decepcionantes. Este tipo de modelos se podrían utilizar también online (*stream learning*) pero en este caso solo se justifica para no procesar los datos en un único bloque.

Una mejor solución ha sido primero preprocesar el conjunto de datos a través de un modelo ipca (PCA incremental) y comprimir los datos en conjuntos de datos mucho más pequeños que caben en la memoria y permiten el uso de una mayor variedad de modelos. Utilizando los primeros 24 componentes se captura el 75 % de la varianza. Ajustando un Random Forest se obtienen buenos resultados (Figura A.4).

La mayoría de las publicaciones sobre el dataset 3W de la bibliografía utilizan modelos offline y obtienen buenos resultados. Por tanto los resultados en este sección del trabajo únicamente confirman que el uso de redes neuronales y Random Forest permite un buen ajuste.

Bibliografía

[1] Nida Aslam, Irfan Ullah Khan, Aisha Alansari, Marah Alrammah, Atheer Alghwairy, Rahaf Alqahtani, Razan Alqahtani, Maryam Almushikes, and Mohammed AL Hashim. Anomaly detection using explainable random forest for the prediction of undesirable events in oil wells. *Applied Computational Intelligence and Soft Computing*, 2022:1558381, Aug 2022.

[2] Manuel Baena-García, José Campo-Ávila, Raúl Fidalgo-Merino, Albert Bifet, Ricard Gavald, and Rafael Morales-Bueno. Early drift detection method. *Fourth International Workshop on Knowledge Discovery from Data Streams*, 01 2006.

[3] Albert Bifet and Ricard Gavaldà. Learning from time-changing data with adaptive windowing. *Proceedings of the 7th SIAM International Conference on Data Mining*, 7, 04 2007.

[4] Albert Bifet, Geoffrey Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: massive online analysis. *Journal of Machine Learning Research*, 11, 05 2010.

[5] Ekaba Bisong. *Batch vs. Online Learning*, pages 199–201. Apress, Berkeley, CA, 2019.

[6] Austin Boyd, Andre Souza, Giovanna Carneiro, Vinicius Machado, Willian Trevizan, Bernardo Coutinho, Paulo Netto, Rodrigo Azeredo, Ralf Polinski, and Andre Bertolini. Presalt carbonate evaluation for santos basin, offshore brazil. *PETROPHYSICS*, 56:577, 12 2015.

[7] Chrisander Brønstad, Sergio L. Netto, and Antonio L.L. Ramos. In *Data-driven Detection and Identification of Undesirable Events in Subsea Oil Wells*, Athene, Greece, 2021. SENSORDEVICES 2021. The Twelfth International Conference on Sensor Device Technologies and Applications.

[8] Bruno Guilherme Carvalho, Ricardo Emanuel Vaz Vargas, Ricardo Menezes Salgado, Celso José Munaro, and Flávio Miguel Varejão. Hyperparameter tuning and feature selection for improving flow instability detection in offshore oil wells. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, pages 1–6, 2021.

[9] Bruno Guilherme Carvalho, Ricardo Emanuel Vaz Vargas, Ricardo Menezes Salgado, Celso José Munaro, and Flávio Miguel Varejão. Flow instability detection in offshore oil wells with multivariate time series machine learning classifiers. In *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*, pages 1–6, 2021.

[10] A. De Salvo Castro, M. De Jesus Rocha Santos, F. Leta, C. Lima, and G. Lima. Unsupervised methods to classify real data from offshore wells. *American Journal of Operations Research*, 11:227–241, 2021.

[11] Ilan Sousa Figueirêdo, Tássio Farias Carvalho, Wenisten José Dantas Silva, Lílian Lefol Nani Guarieiro, and Erick Giovani Sperandio Nascimento. In *Detecting Interesting and Anomalous Patterns In Multivariate Time-Series Data in an Offshore Platform Using Unsupervised Learning*, volume Day 2 Tue, August 17, 2021 of *OTC Offshore Technology Conference*, 08 2021. D021S025R003.

[12] Isvani Frías-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jiménez, Rafael Morales-Bueno, Agustín Ortiz-Díaz, and Yailé Caballero-Mota. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, 2015.

[13] João Gama, Pedro Medas, Gladys Castillo, and Pedro Pereira Rodrigues. Learning with drift detection. In Ana L. C. Bazzan and Sofiane Labidi, editors, *SBIA*, volume 3171 of *Lecture Notes in Computer Science*, pages 286–295. Springer, 2004.

[14] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4), mar 2014.

[15] Rosana Noronha Gemaque, Albert França Josuá Costa, Rafael Giusti, and Eulanda Miranda dos Santos. An overview of unsupervised drift detection methods. *WIREs Data Mining and Knowledge Discovery*, 10(6):e1381, 2020.

[16] Heitor Murilo Gomes, Jesse Read, Albert Bifet, Jean Paul Barddal, and Joao Gama. Machine learning for streaming data: state of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter*, 21:6–22, 11 2019.

[17] Hanqing Hu, Mehmed Kantardzic, and Tegjyot S. Sethi. No free lunch theorem for concept drift detection in streaming data classification: A review. *WIREs Data Mining and Knowledge Discovery*, 10(2):e1327, 2020.

[18] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, page 97–106, New York, NY, USA, 2001. Association for Computing Machinery.

[19] Bartosz Krawczyk, Leandro L. Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.

[20] Jesus L. Lobo, Javier Del Ser, Albert Bifet, and Nikola Kasabov. Spiking neural networks and online learning: An overview and perspectives. *Neural Networks*, 121:88–100, 2020.

[21] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, PP:1–1, 10 2018.

[22] Jesús López Lobo. *New perspectives and methods for stream learning in the presence of concept drift.* PhD thesis, UNIVERSITY OF THE BASQUE COUNTRY UPV/EHU, Spain, 2018.

[23] André Paulo Ferreira Machado, Ricardo Emanuel Vaz Vargas, Patrick Marques Ciarelli, and Celso Jose Munaro. Improving performance of one-class classifiers applied to anomaly detection in oil wells. *Journal of Petroleum Science and Engineering*, 218:110983, 2022.

[24] Matheus A. Marins, Bettina D. Barros, Ismael H. Santos, Daniel C. Barrionuevo, Ricardo E.V. Vargas, Thiago de M. Prego, Amaro A. de Lima, Marcello L.R. de Campos, Eduardo A.B. da Silva, and Sergio L. Netto. Fault detection and classification in oil wells and production/service lines using random forest. *Journal of Petroleum Science and Engineering*, 197:107879, 2021.

[25] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. River: machine learning for streaming data in python, 2020.

[26] National Commission on the BP Deepwater Horizon Oil Spill and Offshore Drilling. *Deep water : the Gulf oil disaster and the future of offshore drilling : report to the President.* Washington, D.C, 2011.

[27] James Reason. Human error: models and management. *BMJ*, 320(7237):768–770, 2000.

[28] Tegjyot Singh Sethi and Mehmed Kantardzic. On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, 82:77–99, 2017.

[29] Alexey Tsymbal. The problem of concept drift: Definitions and related work. 05 2004.

[30] Evren Mert Turan and Johannes Jäschke. Classification of undesirable events in oil well operation. *2021 23rd International Conference on Process Control (PC)*, pages 157–162, 2021.

[31] Ricardo Emanuel Vaz Vargas, Celso José Munaro, Patrick Marques Ciarelli, André Gonçalves Medeiros, Bruno Guberfain do Amaral, Daniel Centurion Barrionuevo, Jean Carlos Dias de Araújo, Jorge Lins Ribeiro, and Lucas Pierezan Magalhães. A realistic and public dataset with rare undesirable real events in oil wells. *Journal of Petroleum Science and Engineering*, 181, 7 2019.

[32] Venkat Venkatasubramanian, Raghunathan Rengaswamy, Kewen Yin, and Surya N. Kavuri. A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. *Computers & Chemical Engineering*, 27(3):293–311, 2003.