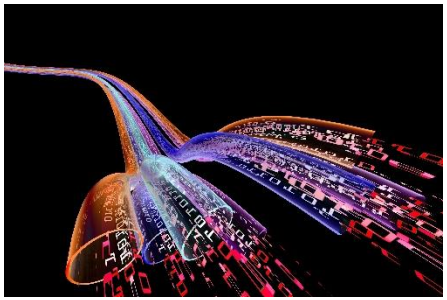


PRÁCTICAS

APLICACIONES AVANZADAS DE LA IA

MÁSTER UNIV. EN INGENIERÍA INFORMÁTICA

Curso: 2025-2026

José Antonio Iglesias Martínez

uc3m

Detección de anomalías en datos de sensores (series temporales)

Flujos de Datos….

Avionica Tests Continuous In-Flight Aircraft Data Streaming

Platform transmits live sensor data to ground analytics during flight operations.

Matt Ryan • Tuesday, February 17, 2026 at 04:22 PM ET Verified Edited By: Zach Vasile

*Avionica ha demostrado una nueva plataforma que **permite transmitir en tiempo real los datos de sensores de una aeronave durante el vuelo a un sistema de análisis en la nube en tierra**, eliminando la necesidad de descargar datos tras el aterrizaje y ofreciendo acceso continuo al estado y rendimiento de la aeronave. Inicialmente integrada con el software de análisis de datos de vuelo basado en la nube de ERGOSS, esta tecnología facilita el monitoreo operativo y la planificación de mantenimiento, y está disponible para su integración en flotas de aerolíneas y con socios de la industria.*

Making the financial case for a data streaming platform in 2026

By Alex Stuart January 29, 2026 Audio mode Dyslexia mode

SUMMARY: Confluent's Alex Stuart explains how data streaming platforms enable organizations to eliminate redundant data rebuilds, accelerate decision-making, and reduce operational costs.

For years, businesses have chased faster, cleaner, more reliable data. That's not new. What *is* new is the speed at which AI has reshaped expectations. In barely a year, organizations have shifted from building systems that keep pace with people to building systems that must keep pace with machines.

This is where a data streaming platform (DSP)

(©your_photo - canva.com)

Las plataformas de data streaming ofrecen un valor económico significativo en 2026 al permitir a las organizaciones eliminar reconstrucciones redundantes de datos y acelerar la disponibilidad de información en tiempo real para procesos críticos, reduciendo así costes operativos y mejorando la rapidez de toma de decisiones. Según el texto, estas plataformas ayudan a las empresas a aprovechar los datos en movimiento para obtener resultados de negocio más ágiles y eficientes, lo que facilita justificar financieramente las inversiones en infraestructuras de streaming frente a enfoques tradicionales basados en almacenamiento y procesamiento por lotes.

OPINION

Accelerating manufacturing with real-time data streams

The integration of IoT, AI, and machine learning is giving birth to smart factories.

by *Rubal Sahni, Area Vice President & Country Manager India - Confluent*

September 4, 2024

SHARE

Expanding the impact of real-time data

IDC analysts predict that 60 per cent of original equipment manufacturers will use AI and real-time asset data to service equipment pre-failure, avoiding 50% of unplanned downtime hours by 2026. Modern factories rely heavily on robots for various tasks. To maximise their efficiency and uptime, real-time data from these robots can be analysed. By using advanced analytics platforms, manufacturers can gain insights into robot performance, predict potential issues, and

But even beyond the factory, real-time data integration enhances supply chain management by improving tracking, forecasting, and inventory control making it possible for businesses to anticipate market trends and align inventory with future demand. Additionally, real-time data streamlines communication across large factories and with external partners, ensuring that everyone stays informed and can make prompt, effective decisions. This leads to greater efficiency, cost savings, and improved performance across manufacturing processes.

Read Next

Enhancing automated material handling with advanced layer gripping technology

Industrial disasters & safety: Enhancing process security in Indian manufacturing

Enhancing vocational training and skill development to meet manufacturing industry needs

Enhancing automated material handling with advanced layer gripping technology

Industrial disasters & safety: Enhancing process security in Indian manufacturing

Enhancing vocational training and skill development to meet manufacturing industry needs

Este artículo enfatiza el papel transformador del streaming de datos en tiempo real en el sector manufacturero. Al capturar y procesar datos a lo largo de toda la cadena de valor de producción, las empresas pueden obtener insights sin precedentes sobre sus operaciones, optimizar procesos y mejorar la calidad del producto.
La integración de tecnologías como IoT, IA y aprendizaje automático está dando lugar a fábricas inteligentes que son más eficientes y ágiles, capaces de responder rápidamente a las demandas cambiantes del mercado.

Flujos de Datos
(Streaming data o Data streams)

Es una secuencia contablemente infinita de elementos.

Flujos de Datos
(Streaming data o Data streams)

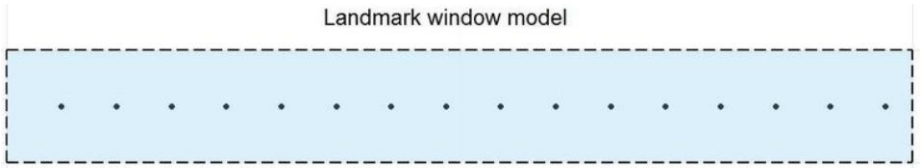
"En el mundo actual, los datos llegan a tasas sin precedentes, y la capacidad de procesar y analizar estos datos a medida que se generan se ha convertido en una ventaja competitiva crucial."

Michael Stonebraker

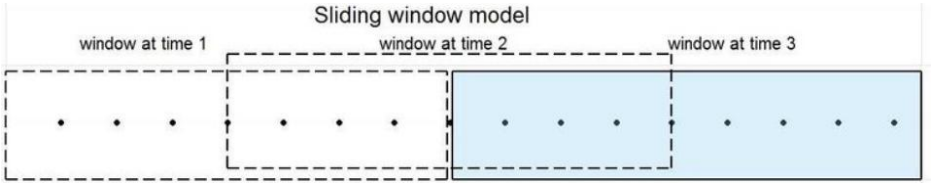
Conceptos relevantes en el flujo de datos:

Técnicas de Ventana

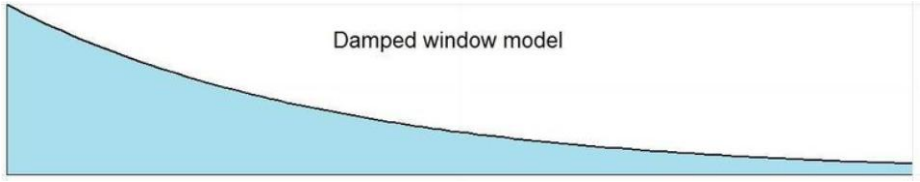
1.



2.



3.



Reference:
https://www.researchgate.net/publication/325977106_An_evaluation_of_data_stream_clustering_algorithms

Conceptos relevantes en el flujo de datos:

Model Drift

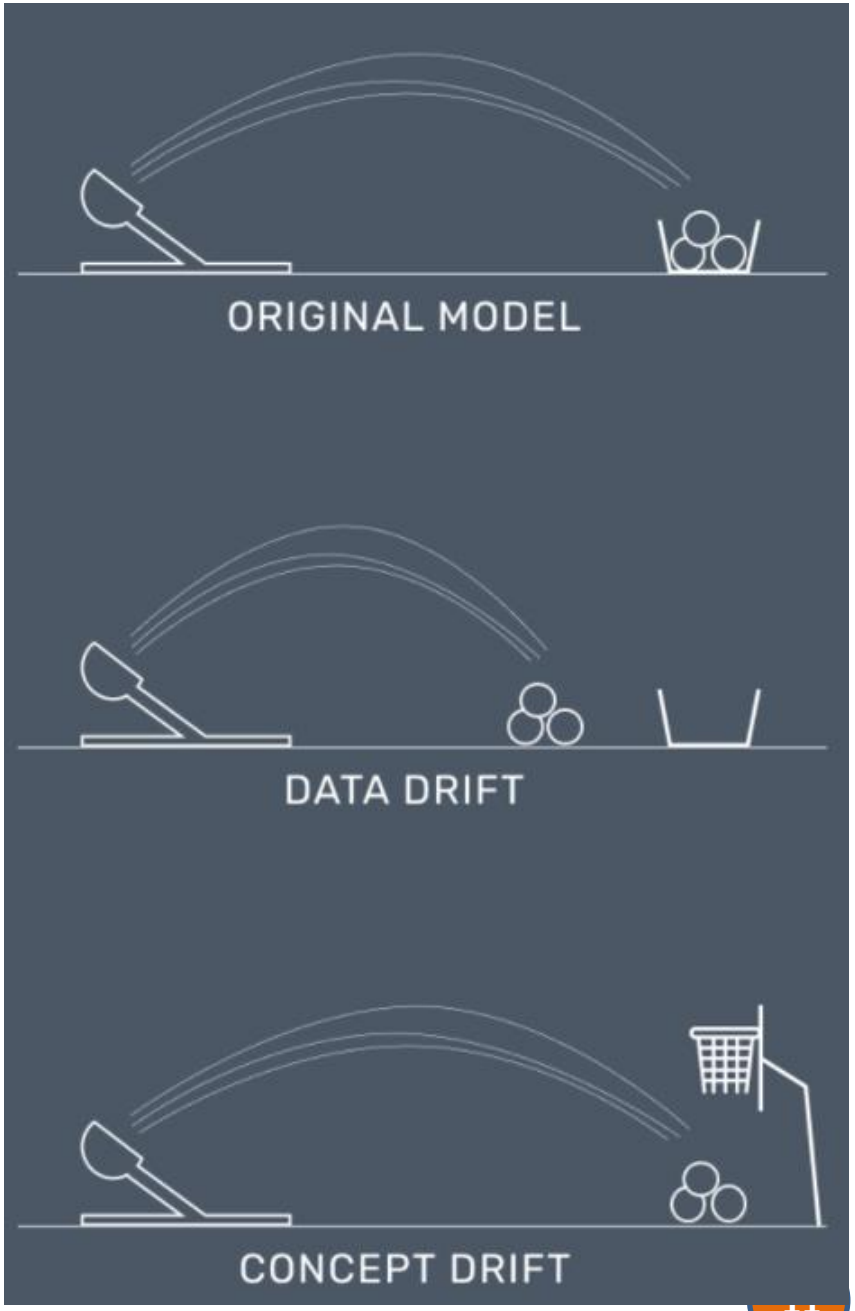
Model Drift

Deterioro de la capacidad de predicción de los modelos como resultado de los cambios en el entorno del mundo real.

Se debe a una serie de razones, como los **cambios** en el entorno digital y los consiguientes cambios en la relación entre las variables.

Hay dos tipos de *Model Drift*:

- *Concept Drift*
- *Data Drift*

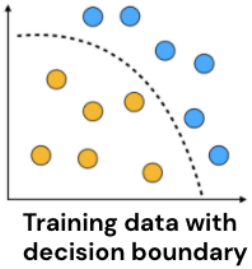


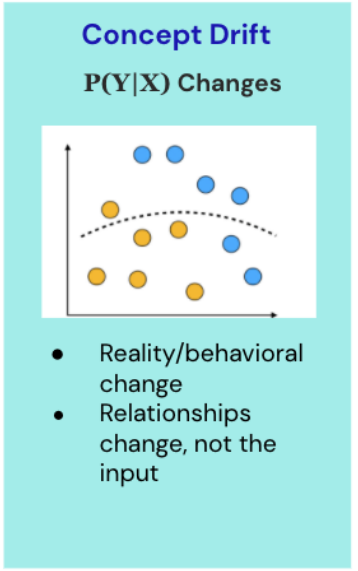
Model Drift – Concept Drift

Situación en la que **las propiedades estadísticas de la variable objetivo** (lo que el modelo intenta predecir) **cambian con el tiempo.**

Es decir, el significado de los datos de entrada con los que se entrenó el modelo ha cambiado significativamente con el tiempo, pero el modelo utilizado no conoce el cambio y, por lo tanto, ya no puede hacer predicciones precisas.

Ejemplo: la definición de spam evoluciona con el tiempo.

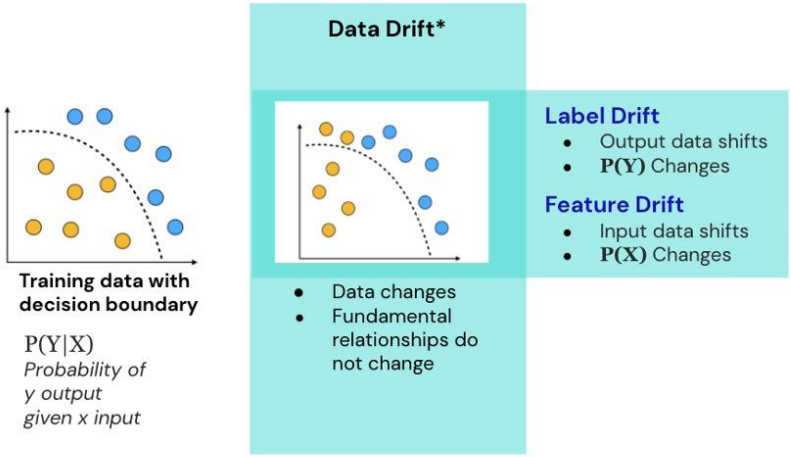




Model Drift – Data Drift

Situación en la que **los datos de entrada han cambiado, por lo que la distribución de las variables es significativamente diferente.** Como resultado, el modelo entrenado no es relevante para estos nuevos datos.

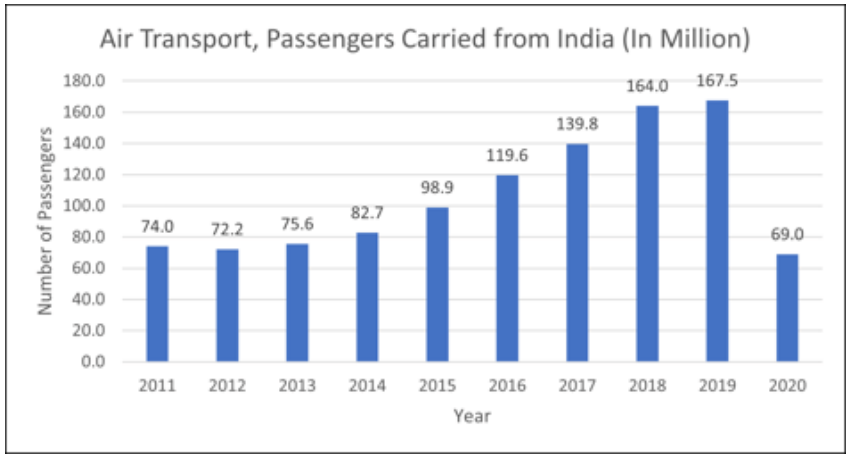
Esto puede ocurrir debido a muchas causas, como el comportamiento estacional o el cambio en la población subyacente

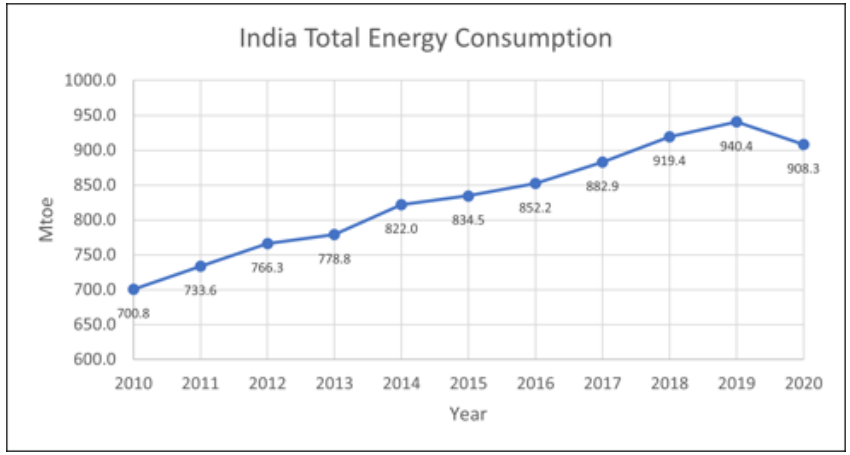


Model Drift – Data Drift

Situación en la que **los datos de entrada han cambiado, por lo que la distribución de las variables es significativamente diferente.** Como resultado, el modelo entrenado no es relevante para estos nuevos datos.

Ejemplo: Cambio en los valores de las características debido a la pandemia.





Herramientas para el manejo de flujos de datos

Herramientas y tecnologías para Machine Learning en Flujos de Datos

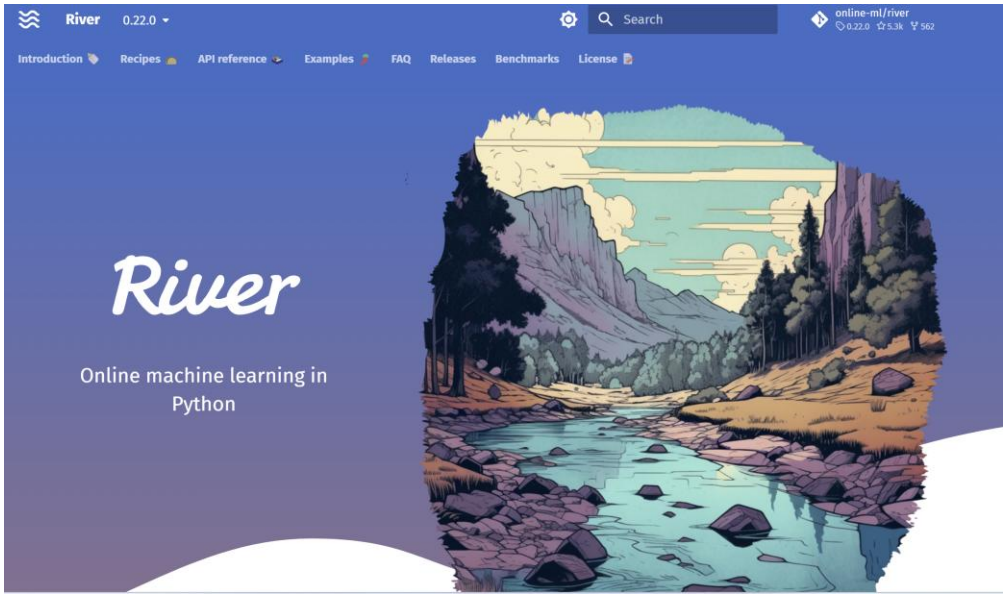
Creme

scikit-multiflow

River is a **Python library** for online machine learning. It is the result of a merger between creme and scikit-multiflow.

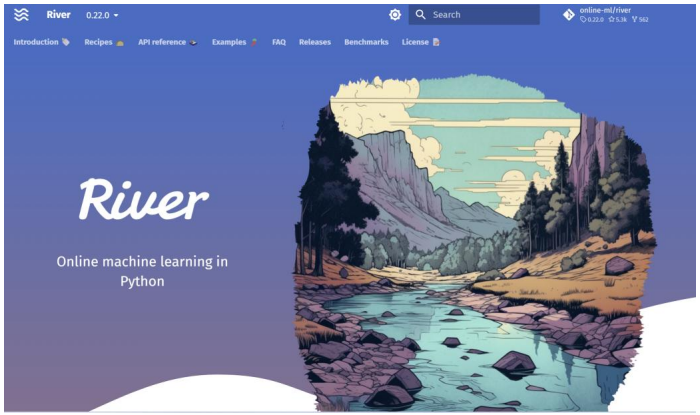
River's ambition is to be the **go-to library for doing machine learning on streaming data.**

Octubre 2022: Versión 0.13.0
Octubre 2023: Versión 0.19.0
Octubre 2024: Versión 0.21.2



https://riverml.xyz/latest/

Herramientas y tecnologías para Machine Learning en Flujos de Datos



Why use River? ¶

Processing one sample at a time

All the tools in the library can be updated with a single observation at a time. They can therefore be used to process streaming data. Depending on your use case, this might be more convenient than using a batch model.

Adapting to drift

In the streaming setting, data can evolve. Adaptive methods are specifically designed to be robust against concept drift in dynamic environments. Many of River's models can cope with concept drift.

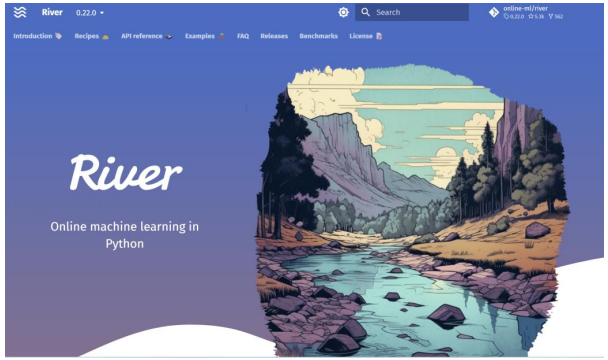
General purpose

River supports different machine learning tasks, including regression, classification, and unsupervised learning. It can also be used for adhoc tasks, such as computing online metrics, as well as concept drift detection.

User experience

River is not the only library allowing you to do online machine learning. But it might just the simplest one to use in the Python ecosystem. River plays nicely with Python dictionaries, therefore making it easy to use in the context of web applications where JSON payloads are aplenty.

Herramientas y tecnologías para Machine Learning en Flujos de Datos



Basic concepts

Here are some concepts to give you a feel for what problems River addresses.

Data streams ¶

River is a library to build online machine learning models. Such models operate on data streams. But a data stream is a bit of a vague concept.

In general, a data stream is a sequence of individual elements. In the case of machine learning, each element is a bunch of features. We call these samples, or observations. Each sample might follow a fixed structure and always contain the same features. But features can also appear and disappear over time. That depends on the use case.

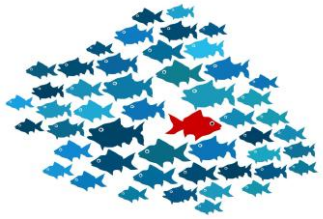
Reactive and proactive data streams

The origin of a data stream can vary, and usually it doesn't matter. You should be able to use River regardless of where your data comes from. It is however important to keep in mind the difference between reactive and proactive data streams.

Reactive data streams are ones where the data comes to you. For instance, when a user visits your website, that's out of your control. You have no influence on the event. It just happens and you have to react to it.

Práctica 4

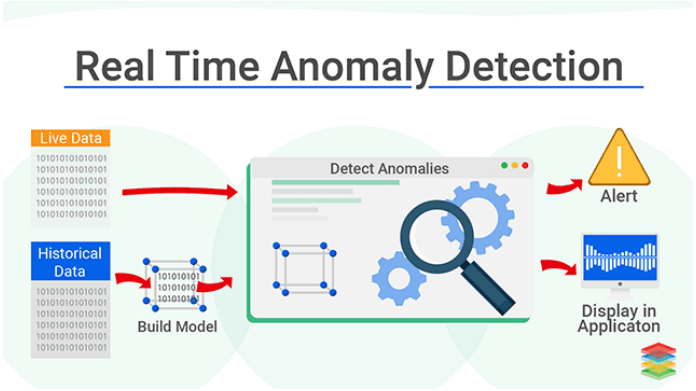
Background de la práctica…

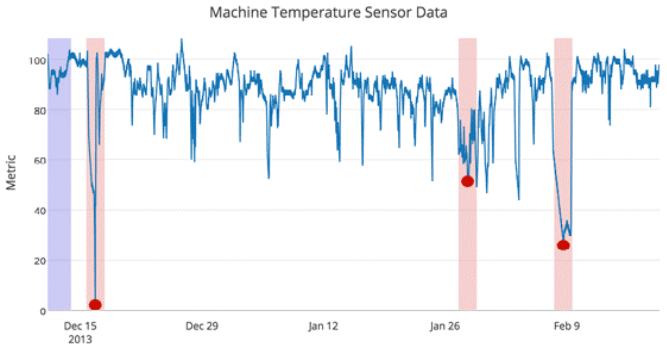


La capacidad de detectar anomalías con antelación y ser capaz de mitigar los riesgos es una capacidad muy valiosa.

Permite además:

- prevenir un tiempo de inactividad no planificado,
- el mantenimiento innecesario,
- gestionar los componentes críticos para estos activos de forma más eficaz.

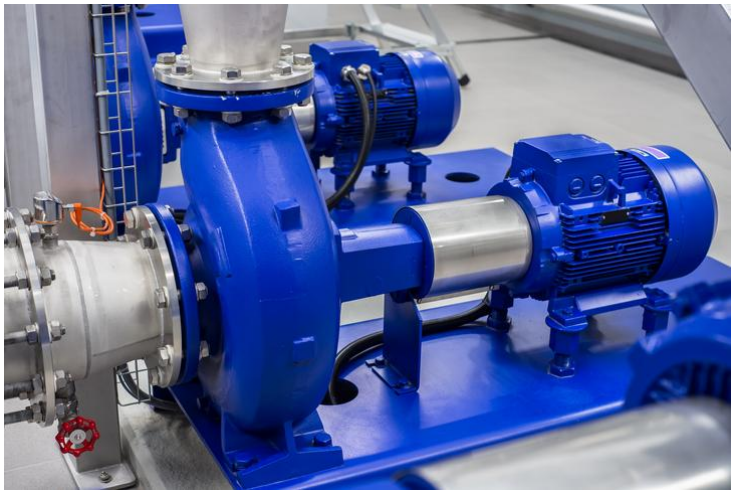




OBJETIVO de la práctica:

El objetivo de esta práctica es la detección de anomalías (*outliers*) de una bomba de agua sensorizada con 52 sensores que toman datos cada minuto. La detección de estas anomalías se realizará en función las más de 220.000 mediciones de cada uno de los sensores**.**

Además, la detección de anomalías deberá realizarse de forma off-line (utilizando todos los datos) y de forma on-line (tratando las mediciones como flujos de datos)



¿Qué es una anomalía en datos de sensores industriales?

En un entorno industrial (como una bomba con 52 sensores), no todas las anomalías son iguales. Es fundamental distinguir el tipo de anomalía para elegir el método adecuado.

1. Point Anomaly (Anomalía puntual)

Un valor individual es anómalo respecto al resto de los datos.

Temperatura habitual: 60–70°C
Lectura puntual: **150°C**

Fácil de detectar con:

- IQR
- Isolation Forest

Detecta errores claros o picos extremos

No captura patrones temporales complejos

¿Qué es una anomalía en datos de sensores industriales?

En un entorno industrial (como una bomba con 52 sensores), no todas las anomalías son iguales. Es fundamental distinguir el tipo de anomalía para elegir el método adecuado.

2. Contextual Anomaly (Anomalía contextual)

Un valor es anómalo dependiendo del contexto temporal.

70°C puede ser normal en verano
70°C puede ser anómalo en invierno

Depende de: Estación, Hora del día, Régimen de funcionamiento, etc

Muy importante en series temporales industriales

Requiere considerar ventanas temporales o modelos dinámicos

¿Qué es una anomalía en datos de sensores industriales?

En un entorno industrial (como una bomba con 52 sensores), no todas las anomalías son iguales. Es fundamental distinguir el tipo de anomalía para elegir el método adecuado.

3. Collective Anomaly (Anomalía colectiva)

Un conjunto de observaciones es anómalo, aunque individualmente no lo sean.
Es decir, ningún valor es extremo por sí solo, pero el patrón completo es anómalo.

Ejemplo en la bomba:
Vibración ligeramente elevada durante 2 horas
Pequeña caída sostenida de presión
Ligero aumento progresivo de temperatura

Individualmente parecen normales, peor en conjunto indican posible fallo incipiente.

Detectable con:

- HST en streaming
- Modelos de ventana deslizante
- Autoencoders

¿Qué es una anomalía en datos de sensores industriales?

En esta práctica:

IQR: detecta principalmente anomalías puntuales.

K-Means: puede capturar desviaciones multivariantes.

HST: puede detectar anomalías más complejas en streaming

Con Ventanas Temporales: Contextuales y colectivas

sensor.csv

Column1	timestamp	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	sensor_05	sensor_06	s
0	01/04/2018 0:00	2465394	4,7092E+16	532118	4,63108E+16	634375	7645975	1341146	
1	01/04/2018 0:01	2465394	4,7092E+16	532118	4,63108E+16	634375	7645975	1341146	
2	01/04/2018 0:02	2444734	4735243	532118	4639757	6388889	7354598	1332465	
3	01/04/2018 0:03	2460474	4,7092E+16	531684	4,63976E+14	628125	7698898	1,33174E+16	
4	01/04/2018 0:04	2445718	4713541	532118	4,63976E+14	6364583	7,6589E+15	1,33536E+16	
5	01/04/2018 0:05	2453588	4,7092E+16	531684	4,63976E+14	6376157	7818568	1341146	
6	01/04/2018 0:06	2455556	4704861	5,31684E+16	4,63976E+14	6333333	7581614	1343316	
7	01/04/2018 0:07	2449653	4713541	5,31684E+16	4,63976E+14	6306713	7577331	1,32523E+16	
8	01/04/2018 0:08	2,46343E+16	4,7092E+16	5,31684E+16	4,63976E+14	6319444	7,45892E+15	1,32885E+16	
9	01/04/2018 0:09	2445718	4717882	531684	4,63976E+14	6417823	7457428	1,33825E+16	
10	01/04/2018 0:10	246441	4748264	53125	4,63976E+14	6377314	7605148	1341146	
11	01/04/2018 0:11	2444734	4791666	531684	4,63976E+14	6356482	7458654	1341146	
12	01/04/2018 0:12	2460474	4826389	53125	4,63976E+14	6300926	7695988	1334635	
13	01/04/2018 0:13	2,44867E+16	484375	531684	4,63976E+14	6386574	756731	1,33174E+16	
14	01/04/2018 0:14	2453588	4856771	531684	4,63976E+14	6324074	8065949	1338976	
15	01/04/2018 0:15	2455556	483941	53125	4639757	6423611	7,81319E+15	1,33536E+16	
16	01/04/2018 0:16	2449653	483941	531684	4,63108E+16	6302084	7789381	1330295	
17	01/04/2018 0:17	2,46343E+16	484809	5,36892E+16	4,63108E+16	6436343	7730572	1334635	
18	01/04/2018 0:18	2445718	4861111	53125	4,63108E+16	6329861	7666199	1334635	
19	01/04/2018 0:19	246441	4861111	531684	4,63108E+16	6443287	7849116	1334635	
20	01/04/2018 0:20	2445718	4908854	5303819	4,63108E+16	6334491	7695741	1334635	
21	01/04/2018 0:21	2460474	4921875	53125	4,63108E+16	6262731	7,87621E+15	1334635	
22	01/04/2018 0:22	2,44867E+16	4878472	53125	4626736	6354166	7626164	1341146	
23	01/04/2018 0:23	2453588	4908854	531684	4,62674E+16	6354166	7925443	1334635	
24	01/04/2018 0:24	2453588	4921875	5303819	4,62674E+16	634375	768876	1,33174E+16	
25	01/04/2018 0:25	2449653	4930555	531684	4,62674E+16	6349537	7576706	1334635	
26	01/04/2018 0:26	2459491	4908854	531684	4,62674E+16	6324074	7563688	1,32885E+16	
27	01/04/2018 0:27	2,44867E+16	4878472	53125	4,62674E+16	6392361	7562278	1349826	
28	01/04/2018 0:28	246441	4835069	531684	4626736	6413195	7583797	1341869	

...

sensor_48	sensor_49	sensor_50	sensor_51	machine_status
1579861	6770834	2430556	2013889	NORMAL
1579861	6770834	2430556	2013889	NORMAL
1559606	6712963	2413194	2037037	NORMAL
1559606	6684028	2404514	203125	NORMAL
1582755	6,65509E+15	2421875	2013889	NORMAL
1646412	6,65509E+15	2416088	2016782	NORMAL
171875	6770834	240162	2002315	NORMAL
1785301	6857639	2413194	2010995	NORMAL
1820023	6944444	2430556	2016782	NORMAL
1866319	697338	2465278	2008102	NORMAL
1909722	7118056	2508681	1999421	NORMAL
1935764	7233796	2531829	2002315	NORMAL
1956019	7233796	2531829	2022569	NORMAL
1964699	7291666	2528935	2028356	NORMAL
2054398	7291666	2520255	2010995	NORMAL
2170139	7378472	2537616	1987847	NORMAL
2291667	7494213	2517361	1979167	NORMAL
239294	7,43634E+15	2465278	1984954	NORMAL
2404514	7349537	2416088	1984954	NORMAL
234375	7291666	2369792	1979167	NORMAL
2213542	7667824	2369792	1987847	NORMAL
2115162	8304398	240162	2013889	NORMAL
2100694	8796296	2436343	2045718	NORMAL
2100694	9230324	2447917	2083333	NORMAL
2074653	9693287	2433449	2109375	NORMAL
2060185	1001157	2447917	2115162	NORMAL
2037037	1018519	2445023	2123843	NORMAL

...

220306	31/08/2018 23:46	2404398	4,76996E+16	5,05642E+16	4,31424E+16	6362673	6706367	150897	1670284	1573351	1508247	4023353
220307	31/08/2018 23:47	2,40046E+16	4,76996E+16	5052083	4,31424E+16	6163195	6495082	1511863	166522	1576968	1,50535E+16	4584744
220308	31/08/2018 23:48	2404398	4,76996E+16	5,05642E+16	4,31424E+16	6,27662E+14	6361317	1511863	166088	1576968	1497396	4230576
220309	31/08/2018 23:49	2,40046E+16	4769965	5,05642E+16	4,31424E+16	6304398	6,26065E+15	1496672	166522	1573351	1506076	4,28191E+16
220310	31/08/2018 23:50	2404398	4761285	5052083	4,31424E+16	6299768	6520339	151548	166522	1,56973E+16	1511863	4303246
220311	31/08/2018 23:51	2402431	4769965	5,05208E+14	4,31424E+16	6,37963E+15	6444035	150897	166088	1585648	1508247	4299954
220312	31/08/2018 23:52	2404398	4761285	5,05208E+14	4,31424E+16	6,37037E+15	6554104	1511863	1661603	1,56178E+16	1511863	4337657
220313	31/08/2018 23:53	2406366	4769965	5052083	4,31424E+16	6,29977E+14	6378558	150463	166522	1,56539E+16	1516204	4237976
220314	31/08/2018 23:54	2396528	4761285	5,05642E+16	4,31424E+16	6,37963E+14	6286275	151331	166088	1,55671E+16	1508247	4301181
220315	31/08/2018 23:55	240735	4769965	5052083	4,31424E+16	6,34722E+14	6459095	1511863	166522	1,56539E+16	1516204	4317085
220316	31/08/2018 23:56	2,40046E+16	4,76996E+16	5,05642E+16	4,31424E+16	6,30903E+14	6583363	151548	1670284	1,56539E+16	1511863	4321038
220317	31/08/2018 23:57	2396528	4,76996E+16	5052083	4,31424E+16	6,25926E+15	6729445	150897	1670284	1,56973E+16	1511863	4312836
220318	31/08/2018 23:58	2406366	4,76996E+16	5,05208E+14	4,31424E+16	6356481	6509175	1511863	1656539	1,57407E+16	1511863	4,23575E+16
220319	31/08/2018 23:59	2396528	4,76996E+16	5,05208E+14	4,31424E+16	6398148	6545634	1511863	166522	1,56539E+16	1,50101E+16	4262814

...

Métodos para la detección de anomalías

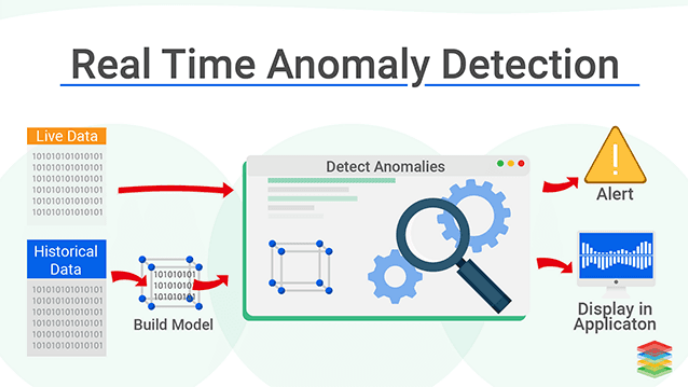
Métodos para la detección de anomalías

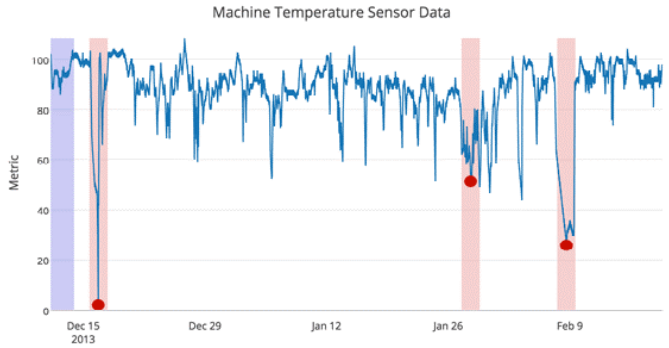
No-Incrementales

- Rango Intercuartil
- K-Means Clustering
- Isolation Forest
- …

Incrementales (Flujos de datos)

- K-Means Clustering Incremental
- Half-Space Trees (HST)
  - Variante Incremental de Isolation Forest.
- …





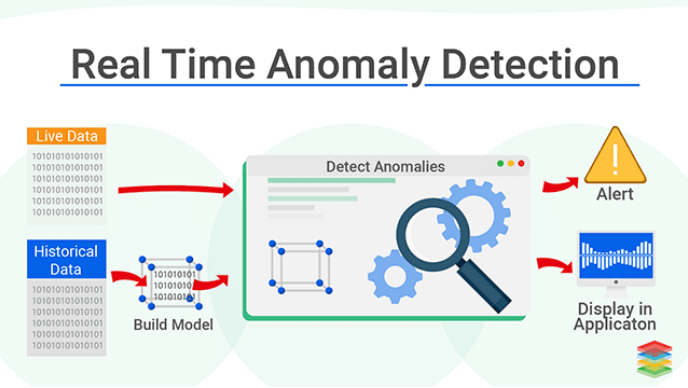
Métodos para la detección de anomalías

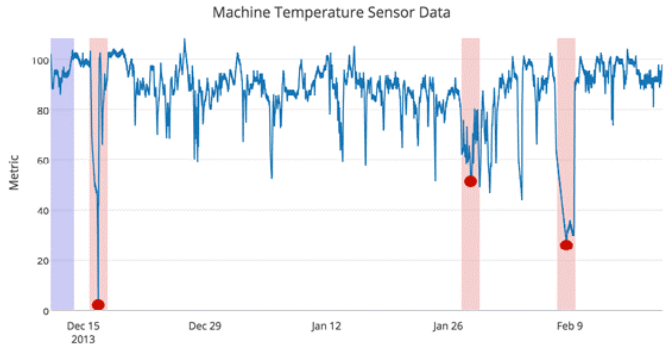
No-Incrementales

- Rango Intercuartil
- K-Means Clustering
- Isolation Forest
- …

Incrementales (Flujos de datos)

- K-Means Clustering Incremental
- Half-Space Trees (HST)
  - Variante Incremental de Isolation Forest.
- …





Rango Intercuartil (IQR)

El IQR mide la dispersión de los datos en el centro de una distribución.

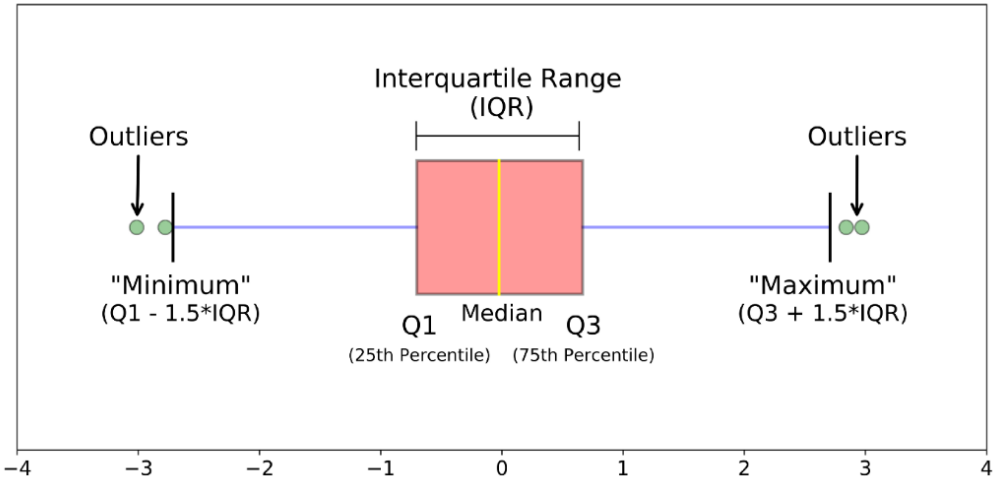
Es la diferencia entre el tercer cuartil (Q3) y el primer cuartil (Q1): **IQR = $Q3 - Q1$**

- Q1 (25%): valor por debajo del cual se encuentra el 25% de los datos
- Q3 (75%): valor por debajo del cual está el 75% de los datos

Uso práctico de IQR es detectar valores atípicos en datos: observaciones que se encuentran:

- por debajo del percentil 25 - 1,5 * IQR, o
- por encima del percentil 75 + 1,5 * IQR

$$\text{Anomalía si: } x < Q1 - 1.5 \times IQR \quad \text{o} \quad x > Q3 + 1.5 \times IQR$$



Rango Intercuartil (IQR)

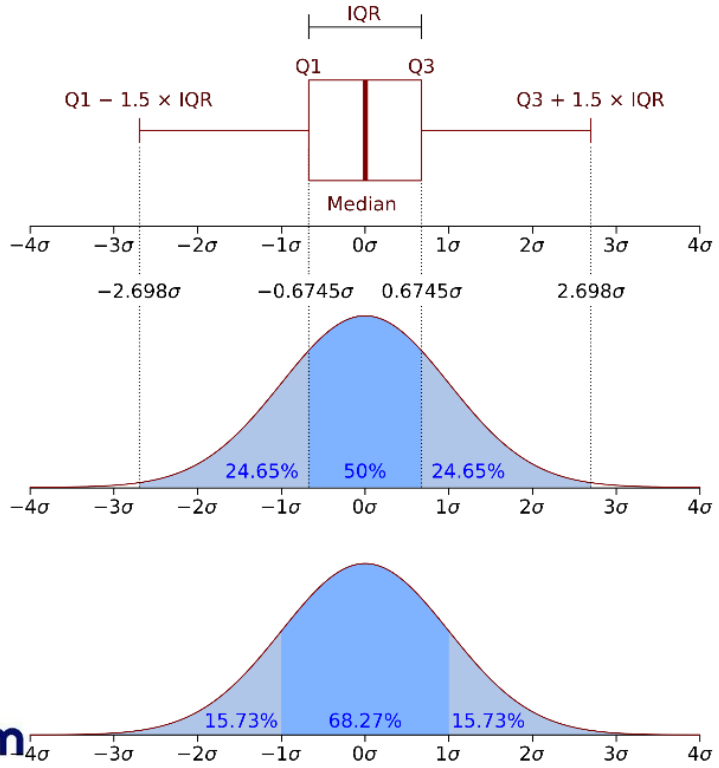
Ejemplo muy sencillo:

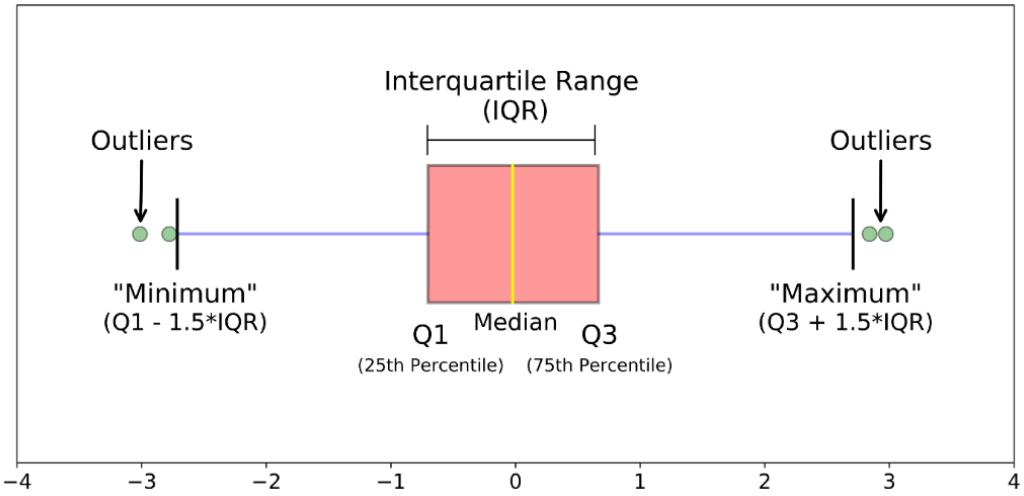
[5, 7, 7, 10, 10, 11, 13, 14]

Q1 = 7 **Q3** = 11 **IQR** = 11 – 7 = 4

- Límite inferior = Q1 - 1.5 × IQR = 8 - 6 = **1**
- Límite superior = Q3 + 1.5 × IQR = 11 + 6 = **17**

Anomalía: **cualquier valor < 1 o > 17**





Rango Intercuartil (IQR)

Limitaciones:

No detecta anomalías multivariantes

- Evalúa cada variable de forma independiente
- No captura combinaciones anómalas entre sensores

Sensible a la fase inicial del stream

En las primeras observaciones:

- Los cuantiles aún no están bien estimados
- El IQR puede ser inestable
- Se pueden generar muchos falsos positivos o negativos

No considera la dimensión temporal

- No detecta anomalías contextuales
- No detecta anomalías colectivas
- No captura tendencias o cambios progresivos

Es simple, rápido y computacionalmente eficiente pero limitado para sistemas industriales multivariantes complejos

Rango Intercuartil (IQR)

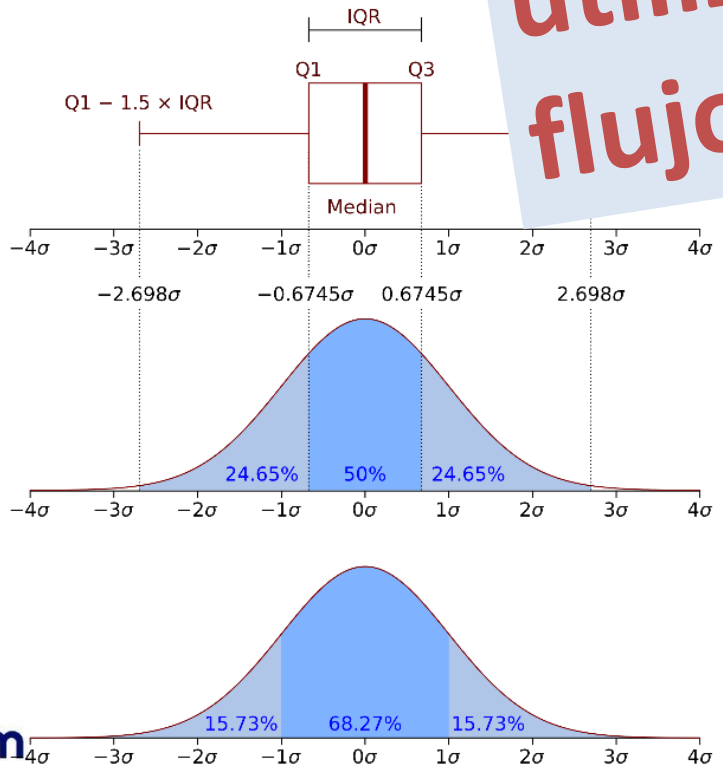
Ejemplo muy sencillo:

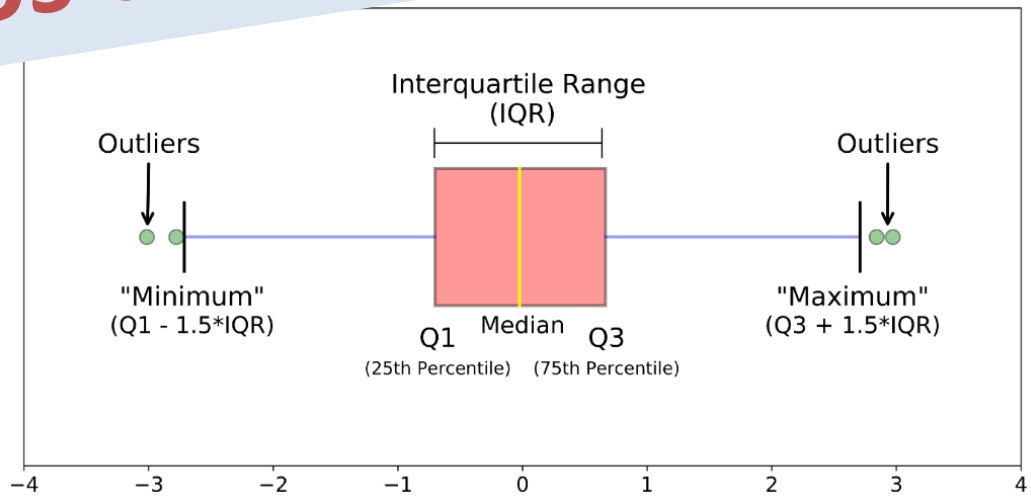
[5, 7, 7, 10, 10, 11, 13, 14]

Q1 = 7 **Q3** = 11 **IQR** = 11 – 7 = 4

- Límite inferior = Q1 - 1.5 × IQR
- Límite superior = Q

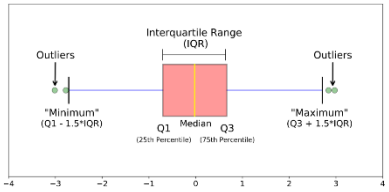
¿Puede utilizarse con flujos de datos?





Rango Intercuartil

River



IQR

API reference

- reco
- rules
- sketch
- stats
  - AbsMax
  - AutoCorr
  - BayesianMean
  - Count
  - Cov
  - EWMean
  - EWVar
  - Entropy
  - IQR
  - Kurtosis
  - Link

Parameters

- **q_inf**

  Default → `0.25`

  Desired inferior quantile, must be between 0 and 1. Defaults to `0.25`.

- **q_sup**

  Default → `0.75`

  Desired superior quantile, must be between 0 and 1. Defaults to `0.75`.

```python
from river import stats

iqr = stats.IQR(q_inf=0.25, q_sup=0.75)

for i in range(0, 1001):
    iqr = iqr.update(i)
    if i % 100 == 0:
        print(iqr.get())
```

Rango Intercuartil

River

¿Puede utilizarse con flujos de datos?

IQR

API reference

- reco
- rules
- sketch
- stats
  - AbsMax
  - AutoCorr
  - BayesianMean
  - Count
  - Cov
  - EWMean
  - EWVar
  - Entropy
  - IQR
  - Kurtosis
  - Link

```
from river import stats

iqr = stats.IQR(q_inf=0.25, q_sup=0.75)

for i in range(0, 1001):
    iqr = iqr.update(i)
    if i % 100 == 0:
        print(iqr.get())
```

```
[1] import numpy as np
    !pip uninstall -y river
    !pip install river~=0.18.0
```

```
from river import stats
from river import datasets

class IQR_Outlier_Detector:
    def __init__(self, factor=1.5):
        self.q1 = stats.Quantile(0.25)  # Primer cuartil
        self.q3 = stats.Quantile(0.75)  # Tercer cuartil
        self.factor = factor

    def update(self, value):
        self.q1.update(value)
        self.q3.update(value)

    def is_outlier(self, value):
        iqr = self.q3.get() - self.q1.get()
        lower_bound = self.q1.get() - self.factor * iqr
        upper_bound = self.q3.get() + self.factor * iqr

        return value < lower_bound or value > upper_bound

# Ejemplo de uso
detector = IQR_Outlier_Detector()


datasetPassengers = datasets.AirlinePassengers()

for value in datasetPassengers:
    detector.update(value[1])
    if detector.is_outlier(value[1]):
        print(f"{value[1]} is an outlier!")
```

```
135 is an outlier!
148 is an outlier!
```

Rango Intercuartil

River

¿Puede utilizarse con flujos de datos?

```
from river import stats
from river import datasets

class IQR_Outlier_Detector:
    def __init__(self, factor=1.5):
        self.q1 = stats.Quantile(0.25)  # Primer cuartil
        self.q3 = stats.Quantile(0.75)  # Tercer cuartil
        self.factor = factor

    def update(self, value):
        self.q1.update(value)
        self.q3.update(value)

    def is_outlier(self, value):
        iqr = self.q3.get() - self.q1.get()
        lower_bound = self.q1.get() - self.factor * iqr
        upper_bound = self.q3.get() + self.factor * iqr

        return value < lower_bound or value > upper_bound
```

```
# Ejemplo de uso
detector = IQR_Outlier_Detector()


datasetPassengers = datasets.AirlinePassengers()

for value in datasetPassengers:
    detector.update(value[1])
    if detector.is_outlier(value[1]):
        print(f"{value[1]} is an outlier!")
```

Rango Intercuartil

River

¿Puede utilizarse con flujos de datos?

```
import pandas as pd
from river import stats

# Se carga el dataset que se va a utilizar:
file_path = './creditcard.csv'
df = pd.read_csv(file_path)

# Crear diccionario para almacenar los cuantiles (Q1 y Q3) para cada característica numérica
iqr_stats = {col: (stats.Quantile(0.25), stats.Quantile(0.75)) for col in df.select_dtypes(include=['float64', 'int64']).columns}

# Limitar el rango intercuartil: el rango "normal" suele considerarse dentro de 1.5 veces el IQR
iqr_factor = 1.5

# Inicializar un diccionario para contar las anomalías detectadas
anomalies = {col: 0 for col in iqr_stats.keys()}

# Función para verificar si un punto es una anomalía basándose en en IQR
def is_anomaly(val, q1, q3):
    iqr = q3 - q1
    lower_bound = q1 - iqr_factor * iqr
    upper_bound = q3 + iqr_factor * iqr
    return val < lower_bound or val > upper_bound
```

```
# Iterar sobre cada fila de los datos como si fuera un flujo de datos
for _, row in df.iterrows():
    for col, (q1_stat, q3_stat) in iqr_stats.items():
        val = row[col]

        # Actualizar los cuantiles de Q1 y Q3 de manera incremental
        q1_stat.update(val)
        q3_stat.update(val)

        # Obtener los valores actuales de Q1 y Q3
        q1 = q1_stat.get()
        q3 = q3_stat.get()

        # Comprobar que tanto q1 como q3 no son None
        if q1 is not None and q3 is not None:
            # Verificar si el valor es una anomalía
            if is_anomaly(val, q1, q3):
                anomalies[col] += 1
                print(f"Anomalía detectada en {col} con valor {val}")

# Mostrar el número total de anomalías detectadas por columna
print("Anomalías detectadas por columna:")
print(anomalies)
```

Rango Intercuartil

River

¿Puede utilizarse con flujos de datos?

```
import pandas as pd
from river import stats

# Se carga el dataset que se va a utilizar:
file_path = './creditcard.csv'
df = pd.read_csv(file_path)

# Crear diccionario para almacenar los cuantiles (Q1 y Q3) para cada característica numérica
iqr_stats = {col: (stats.Quantile(0.25), stats.Quantile(0.75)) for col in df.select_dtypes(include=['float64', 'int64']).columns}

# Limitar el rango intercuartil: el rango "normal" suele considerarse dentro de 1.5 veces el IQR
iqr_factor = 1.5

# Inicializar un diccionario para contar las anomalías detectadas
anomalies = {col: 0 for col in iqr_stats.keys()}

# Función para verificar si un punto es una anomalía basándose en en IQR
def is_anomaly(val, q1, q3):
    iqr = q3 - q1
    lower_bound = q1 - iqr_factor * iqr
    upper_bound = q3 + iqr_factor * iqr
    return val < lower_bound or val > upper_bound
```

```
# Iterar sobre cada fila de los datos como si fuera un flujo de datos
for _, row in df.iterrows():
    for col, (q1_stat, q3_stat) in iqr_stats.items():
        val = row[col]

        # Actualizar los cuantiles de Q1 y Q3 de manera incremental
        q1_stat.update(val)
        q3_stat.update(val)

        # Obtener los valores actuales de Q1 y Q3
        q1 = q1_stat.get()
        q3 = q3_stat.get()

        # Comprobar que tanto q1 como q3 no son None
        if q1 is not None and q3 is not None:
            # Verificar si el valor es una anomalía
            if is_anomaly(val, q1, q3):
                anomalies[col] += 1
                print(f"Anomalía detectada en {col} con valor {val}")

# Mostrar el número total de anomalías detectadas por columna
print("Anomalías detectadas por columna:")
print(anomalies)
```

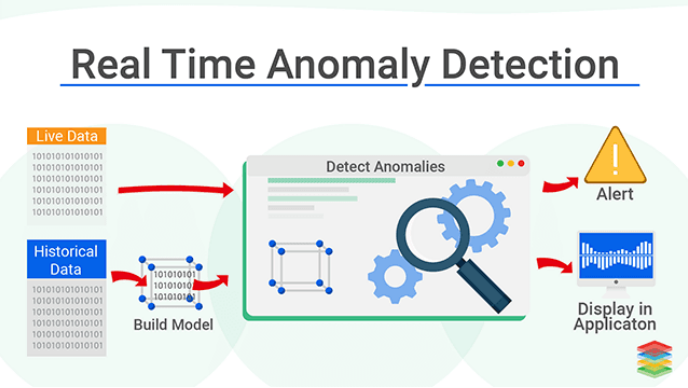
Métodos para la detección de anomalías

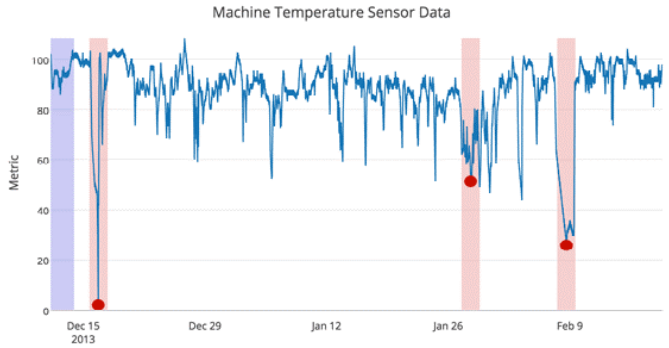
No-Incrementales

- Rango Intercuartil
- K-Means Clustering
- Isolation Forest
- …

Incrementales (Flujos de datos)

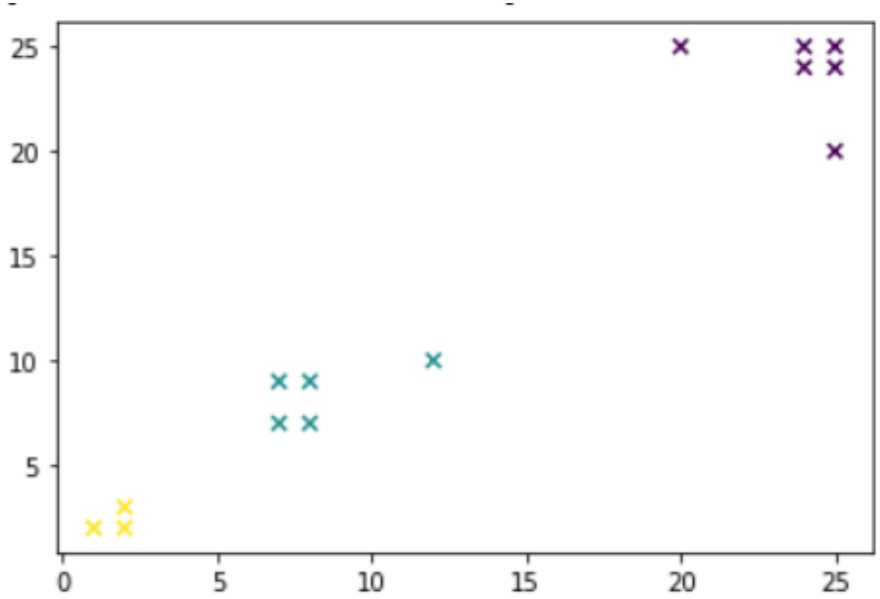
- K-Means Clustering Incremental
- Half-Space Trees (HST)
  - Variante Incremental de Isolation Forest.
- …





K-Means Clustering

Calcular la distancia entre cada punto y su centroide más cercano.
Las distancias más grandes se consideran anomalías.

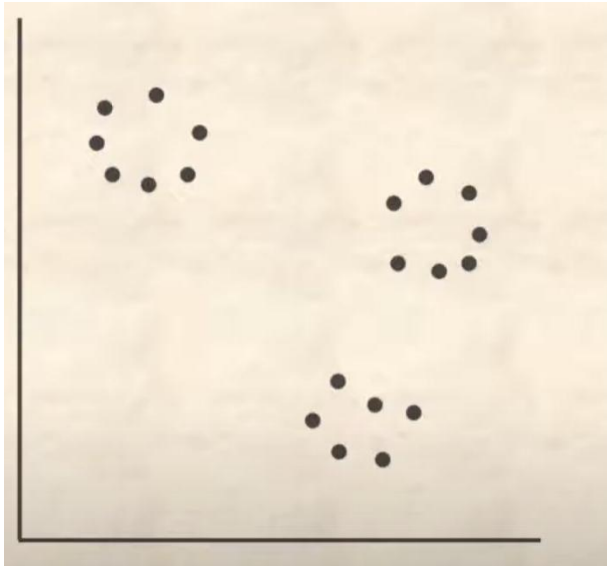


Las distancias pueden ser calculadas utilizando una distancia 'euclídea' o cualquier otro tipo de distancia adecuada. Después de obtener las distancias de los puntos, se define la ratio umbral como percentil para encontrar los valores atípicos. Cuando se decide una ratio umbral, se están ordenando todas las distancias de todos los puntos (a sus propios centros) y se consideran como atípicos aquellos que superan un percentil.

K-Means Clustering

Algoritmo de Lloyd para K-Means

Lloyd es la implementación más conocida de k-Means:

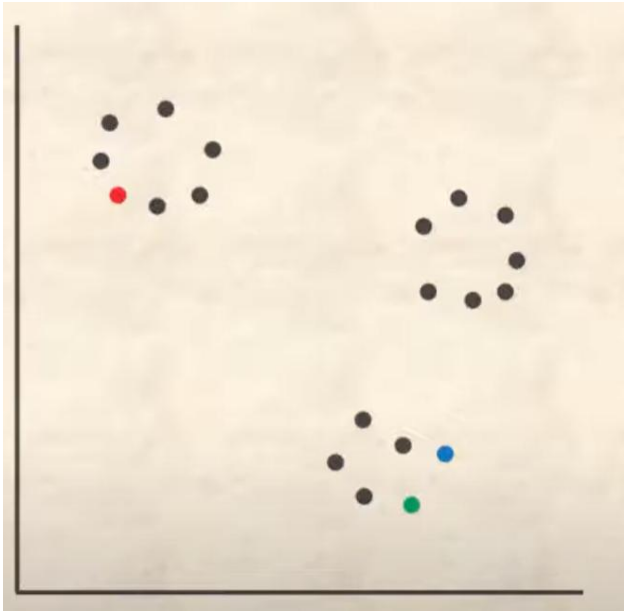


K-Means Clustering

Algoritmo de Lloyd para K-Means

Lloyd es la implementación más conocida de k-Means:

- Se eligen inicialmente k puntos de datos como centroides, al azar.

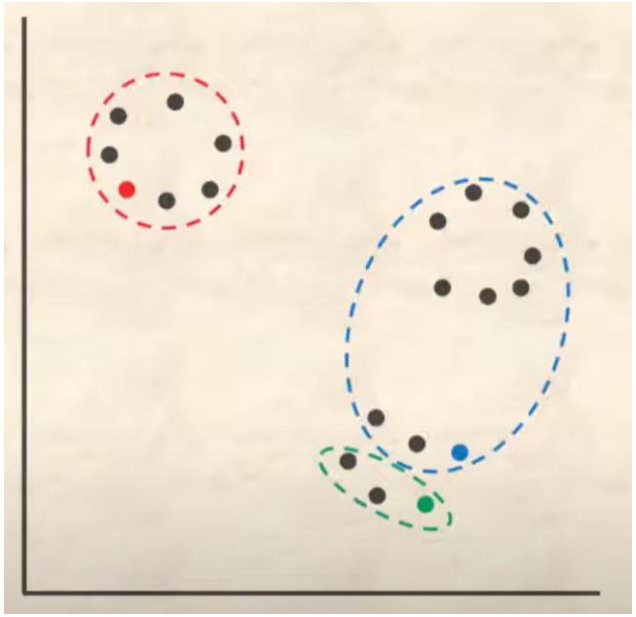


K-Means Clustering

Algoritmo de Lloyd para K-Means

Lloyd es la implementación más conocida de k-Means:

- Se eligen inicialmente k puntos de datos como centroides, al azar.
- Todos los puntos de datos se asignan a su centroide más cercano.

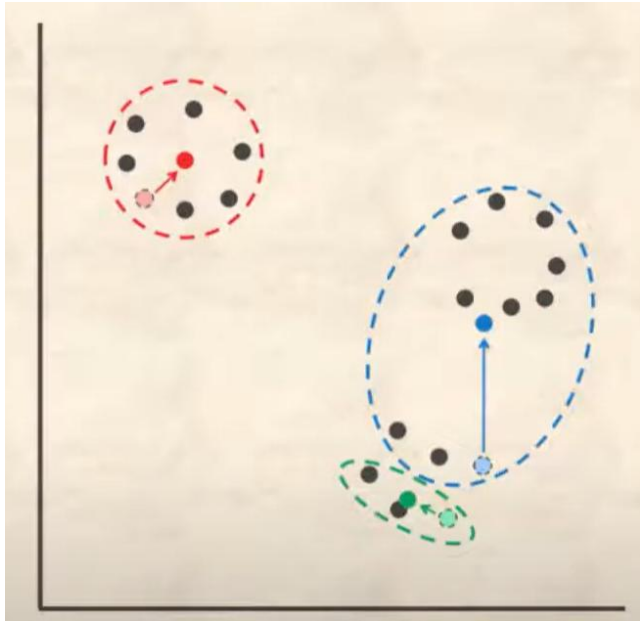


K-Means Clustering

Algoritmo de Lloyd para K-Means

Lloyd es la implementación más conocida de k-Means:

- Se eligen inicialmente k puntos de datos como centroides, al azar.
- Todos los puntos de datos se asignan a su centroide más cercano.
- Se recalculan los nuevos centroides teniendo en cuenta los centros de gravedad (aquél punto con menor distancia acumulada al resto de datos de su cluster) todos los clústeres.

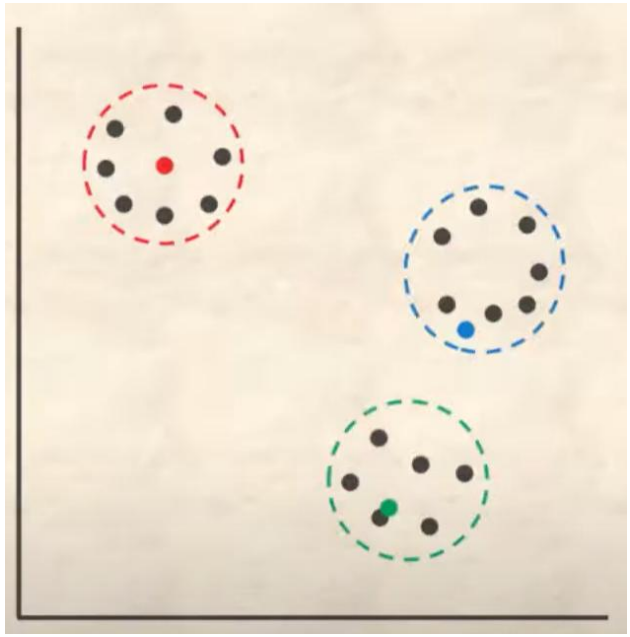


K-Means Clustering

Algoritmo de Lloyd para K-Means

Lloyd es la implementación más conocida de k-Means:

- Se eligen inicialmente k puntos de datos como centroides, al azar.
- Todos los puntos de datos se asignan a su centroide más cercano.
- Se recalculan los nuevos centroides teniendo en cuenta los centros de gravedad (aquél punto con menor distancia acumulada al resto de datos de su cluster) todos los clústeres.
- Se asigna cada punto al centro más cercano

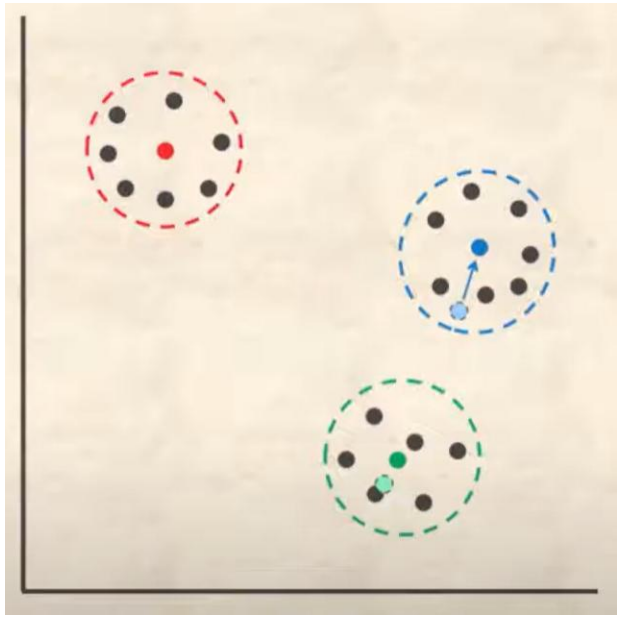


K-Means Clustering

Algoritmo de Lloyd para K-Means

Lloyd es la implementación más conocida de k-Means:

- Se eligen inicialmente k puntos de datos como centroides, al azar.
- Todos los puntos de datos se asignan a su centroide más cercano.
- Se recalculan los nuevos centroides teniendo en cuenta los centros de gravedad (aquél punto con menor distancia acumulada al resto de datos de su cluster) todos los clústeres.
- Se asigna cada punto al centro más cercano



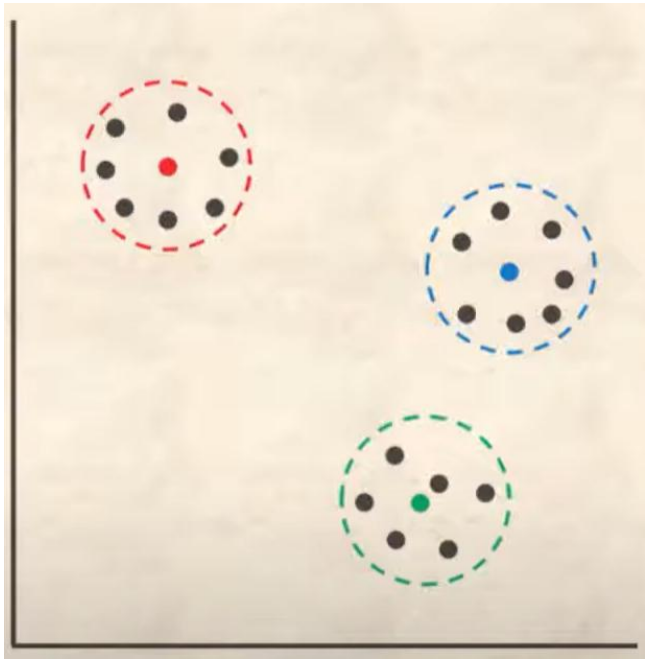
K-Means Clustering

Algoritmo de Lloyd para K-Means

Lloyd es la implementación más conocida de k-Means:

- Se eligen inicialmente k puntos de datos como centroides, al azar.
- Todos los puntos de datos se asignan a su centroide más cercano.
- Se recalculan los nuevos centroides teniendo en cuenta los centros de gravedad (aquél punto con menor distancia acumulada al resto de datos de su cluster) todos los clústeres.
- Se asigna cada punto al centro más cercano

Incompatible con flujos de datos!

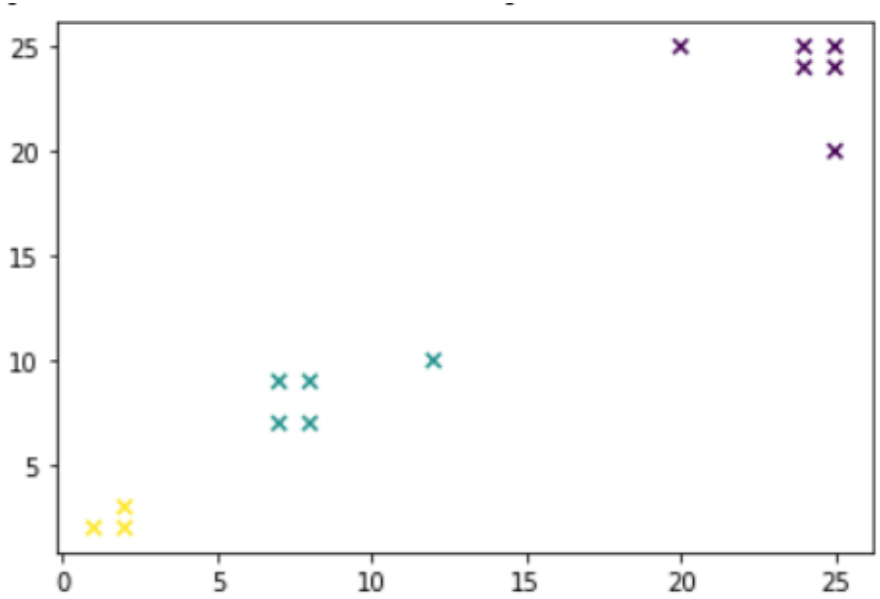


Se repite hasta que los centroides no varíen.

K-Means Clustering

¿Puede utilizarse con flujos de datos?

Calcular la distancia entre cada punto y su centroide más cercano.
Las distancias más grandes se consideran anomalías.



Las distancias pueden ser calculadas utilizando una distancia 'euclídea' o cualquier otro tipo de distancia adecuada. Después de obtener las distancias de los puntos, se define la ratio umbral como percentil para encontrar los valores atípicos. Cuando se decide una ratio umbral, se están ordenando todas las distancias de todos los puntos (a sus propios centros) y se consideran como atípicos aquellos que superan un percentil.

K-Means Clustering Incremental

Para aplicarse con un flujo de datos:

- Se recibe un nueva observación (del flujo de datos).
- Se le asigna el clúster más cercano a dicha observación.
- Se mueve la posición central del clúster hacia la nueva observación.

El parámetro *halflife* determina en qué medida se mueve el cluster hacia la nueva observación.

K-Means Clustering Incremental

Parámetros:

n_clusters - por defecto 5
Número máximo de clusters a asignar.

halflife - por defecto 0.5
Determina en qué medida hay que desplazar los centros de los clusters (centroides) hacia la nueva observación. Valor entre 0 y 1.

mu - por defecto 0
Media de la distribución normal utilizada para instanciar las posiciones de los clusters.

sigma - por defecto 1
Desviación estándar de la distribución normal utilizada para instanciar las posiciones de los clusters.

p - Por defecto, 2
Parámetro de potencia para la métrica de medidas de distancia a utilizar: Cuando p=1, corresponde a la distancia Manhattan, mientras que p=2 corresponde a la distancia euclidiana.

seed (int) - Por defecto, None
Semilla aleatoria utilizada para generar las posiciones iniciales del centroide.

K-Means Clustering Incremental

Aspectos a considerar:

Elección de hiperparámetros

- **Número de clusters (n_clusters)**

  Elegir k sin justificación puede producir:

  - Sobreajuste, Detección inestable de anomalías o Clusters no interpretables

- **Umbral fijo (threshold = 60)**

  Un umbral fijo:

  - No se adapta a cambios en el sistema
  - Puede generar falsos positivos en caso de drift

Impacto del parámetro *halflife*

- El parámetro **halflife** controla cuánto se mueve el centroide hacia la nueva observación.
  - Valores altos → adaptación rápida (más sensible)
  - Valores bajos → mayor estabilidad (menos reactivo)

Mejor alternativa: Umbral adaptativo

En lugar de un umbral fijo: Percentil dinámico (Marcar como anomalía el top 1% o 5% de distancias) o Media + k·desviación estándar. Se adapta automáticamente a cambios de distribución y hace al sistema más robusto frente a drift

K-Means Clustering Incremental

```
!pip install river
```

```
import river
```

```
from river import datasets
from river.datasets import synth
```

```
datasetSinteticoSEA = synth.SEA()
```

```
datasetSintetico = datasetSinteticoSEA.take(100)
```

```
from river import cluster
k_means = cluster.KMeans(n_clusters=3, halflife=0.4, sigma=3, seed=0)
```

```
for dato,y in datasetSintetico:
  k_means = k_means.learn_one(dato)
  pred = k_means.predict_one(dato)
  print("Observación [",dato[0],",",dato[0],",",dato[0],"] asignada al cluster: ",pred)
```

cluster.KMeans

River

K-Means Clustering Incremental

```
import pandas as pd
from river import cluster
import math

# Cargar el archivo creditcard.csv en un DataFrame usando pandas
file_path = './creditcard.csv'
df = pd.read_csv(file_path)

# Inicializar K-Means con 2 clústeres (puedes ajustar este valor según tus datos)
kmeans = cluster.KMeans(n_clusters=2, seed=42)

# Umbral para considerar un punto como anomalía basándonos en la distancia al centroide
threshold = 60.0

# Para almacenar el conteo de anomalías
anomalies_detected = 0

# Función para calcular la distancia euclidiana entre un punto y un centroide
def euclidean_distance(point, center):
    return math.sqrt(sum((point[feature] - center[feature]) ** 2 for feature in point))
        return value < lower_bound or value > upper_bound
```

```
# Iterar sobre cada fila del DataFrame para simular un flujo de datos
for _, row in df.iterrows():
    # Seleccionar todas las columnas numéricas excepto 'Time' y 'Class'
    features = row.drop(labels=['Time', 'Class']).to_dict()

    # Predecir el clúster más cercano
    cluster_id = kmeans.predict_one(features)

    # Obtener los centroides aprendidos hasta el momento
    centroids = kmeans.centers

    # Si el modelo ha aprendido suficientes centroides, calcular la distancia
    if centroids:
        # Obtener el centroide del clúster asignado
        centroid = centroids[cluster_id]

        # Calcular la distancia euclidiana al centroide
        distance_to_centroid = euclidean_distance(features, centroid)

        # Si la distancia es mayor que el umbral, lo consideramos una anomalía
        if distance_to_centroid > threshold:
            anomalies_detected += 1
            print(f"Anomalía detectada en el punto: {row['Time']} con distancia {distance_to_centroid:.2f}")

    # Actualizar el modelo KMeans con el nuevo punto
    kmeans.learn_one(features)

# Mostrar el número total de anomalías detectadas
print(f"Total de anomalías detectadas: {anomalies_detected}")
```

cluster.KMeans

River

K-Means Clustering Incremental

```
import pandas as pd
from river import cluster, preprocessing
import math

# Cargar el archivo creditcard.csv en un DataFrame usando pandas
file_path = './creditcard.csv'
df = pd.read_csv(file_path)

# Crear un scaler para estandarizar los datos
scaler = preprocessing.StandardScaler()

# Inicializar K-Means con 2 clústeres (puedes ajustar este valor según tus datos)
kmeans = cluster.KMeans(n_clusters=5, seed=42)

# Umbral para considerar un punto como anomalía basándonos en la distancia al centroide
threshold = 60.0

# Para almacenar el conteo de anomalías
anomalies_detected = 0

# Función para calcular la distancia euclidiana entre un punto y un centroide
def euclidean_distance(point, center):
    return math.sqrt(sum((point[feature] - center[feature]) ** 2 for feature in point))
```

```
# Iterar sobre cada fila del DataFrame para simular un flujo de datos
for _, row in df.iterrows():
    # Seleccionar todas las columnas numéricas excepto 'Time' y 'Class'
    features = row.drop(labels=['Time', 'Class']).to_dict()

    # Actualizar el escalador con los nuevos datos
    scaler.learn_one(features)

    # Estandarizar las características
    features = scaler.transform_one(features)

    # Predecir el clúster más cercano
    cluster_id = kmeans.predict_one(features)
    # Obtener los centroides aprendidos hasta el momento
    centroids = kmeans.centers

    # Si el modelo ha aprendido suficientes centroides, calcular la distancia
    if centroids:
        # Obtener el centroide del clúster asignado
        centroid = centroids[cluster_id]
        # Calcular la distancia euclidiana al centroide
        distance_to_centroid = euclidean_distance(features, centroid)
        # Si la distancia es mayor que el umbral, lo consideramos una anomalía
        if distance_to_centroid > threshold:
            anomalies_detected += 1
            print(f"Anomalía detectada en el punto: {row['Time']} con distancia {distance_to_centroid:.2f}")

    # Actualizar el modelo KMeans con el nuevo punto
    kmeans.learn_one(features)

# Mostrar el número total de anomalías detectadas
print(f"Total de anomalías detectadas: {anomalies_detected}")
```

Necesario porque las diferencias en las escalas de las variables pueden afectar los resultados. Así, Las variables pueden ser comparadas directamente, incluso si tienen unidades diferentes o distribuciones con escalas distintas.

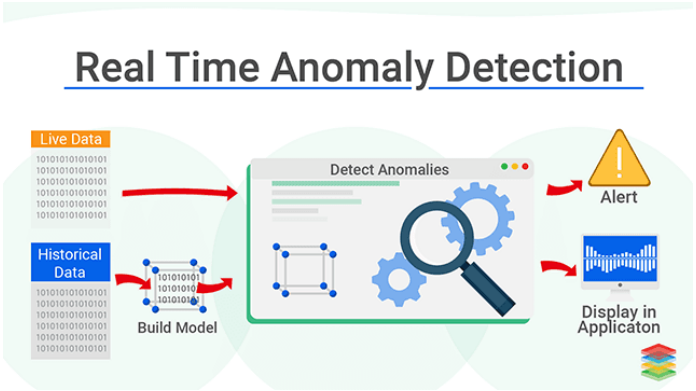
Métodos para la detección de anomalías

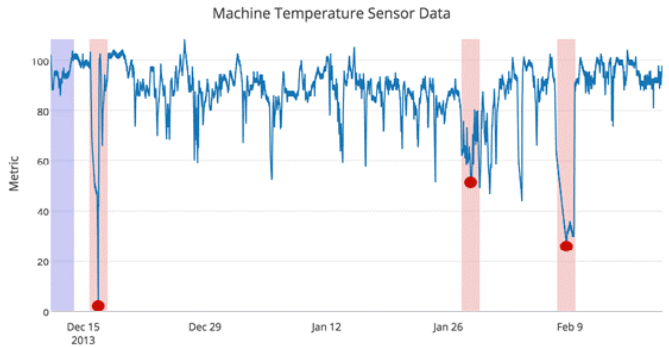
No-Incrementales

- Rango Intercuartil
- K-Means Clustering
- **Isolation Forest**
- …

Incrementales (Flujos de datos)

- K-Means Clustering Incremental
- Half-Space Trees (HST)
  - Variante Incremental de Isolation Forest.
- …





Isolation Forest

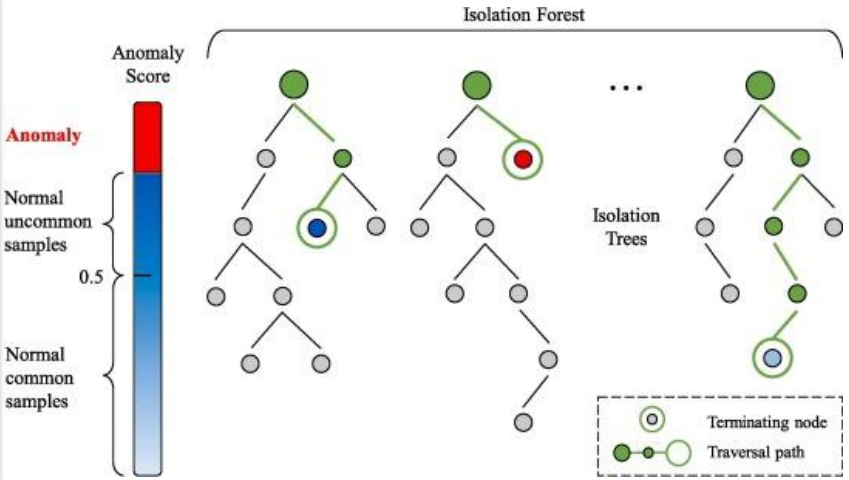
¿Puede utilizarse con flujos de datos?

Método no supervisado para **identificar anomalías (outliers)** cuando los **datos no están etiquetados**, es decir, no se conoce la clasificación real (anomalía - no anomalía) de las observaciones.

Su funcionamiento está inspirado en el algoritmo de clasificación y regresión ***Random Forest***.

Al igual que en *Random Forest*, está formado por la combinación de múltiples árboles llamados ***isolation tres***.

Sin embargo, en los isolation tree, la selección de los puntos de división se hace de forma aleatoria. **Aquellas observaciones con características distintas al resto quedarán aisladas a las pocas divisiones**, por lo que el número de nodos necesarios para llegar a esta observación desde el inicio del árbol (profundidad) es menor que para el resto.



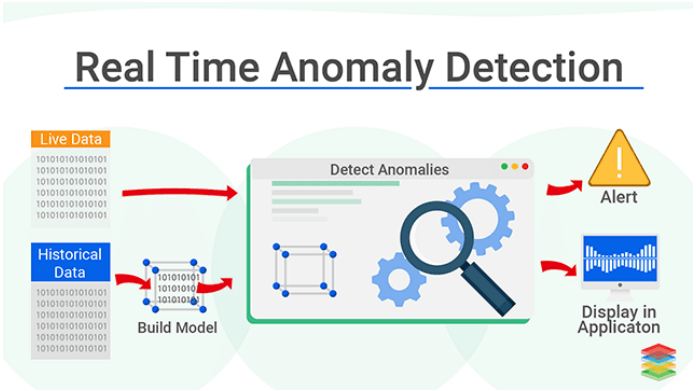
Métodos para la detección de anomalías

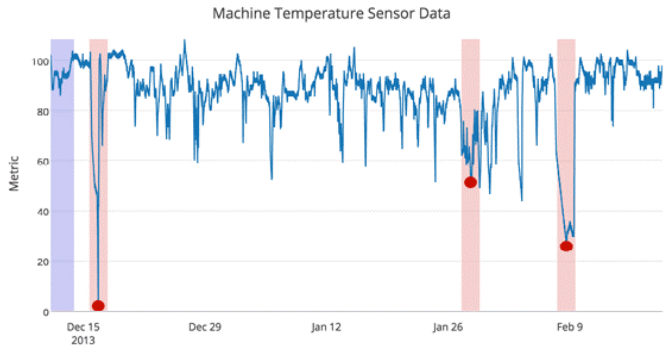
No-Incrementales

- Rango Intercuartil
- K-Means Clustering
- **Isolation Forest**
- …

Incrementales (Flujos de datos)

- K-Means Clustering Incremental
- **Half-Space Trees (HST)**
  - Variante Incremental de Isolation Forest.
- …





Half-space trees (HST)

Variante incremental de Isolation Forest

Por defecto, la implementación de HST en River asume que **cada característica tiene valores que están comprendidos entre 0 y 1**. Si este no fuera así, se debe especificar manualmente los límites a través del argumento límites. Si no conoce los límites de antemano, entonces puede utilizar un ***preprocesamiento.MinMaxScaler*** como un paso inicial de preprocesamiento (considerarlo en un pipeline).

La implementación actual construye los árboles la primera vez que se llama al método **learn_one.** Por lo tanto, la primera llamada a **learn_one** puede ser lenta, mientras que las llamadas posteriores serán muy rápidas en comparación.
En general, el tiempo de cálculo tanto de learn_one como de score_one escala linealmente con el número de árboles, y exponencialmente con la altura de cada árbol.

Puntuaciones altas indican anomalías. Puntuaciones bajas indican observaciones normales.

Half-space trees (HST)

Variante incremental de Isolation Forest

River

anomaly.HalfSpaceTrees

Methods

learn_one

score_one

Parámetros:

n_trees - por defecto 10
Número de árboles a utilizar.

height - por defecto 8
Altura de cada árbol. Un árbol de altura h se compone de h + 1 niveles.

Window_size- por defecto 250
Número de observaciones a utilizar para calcular los valores de cada nodo de cada árbol.

limits (Dict[Hashable, Tuple[float, float]]) - por defecto None
Especifica el rango de cada característica. Por defecto se asume que cada característica está en el rango [0, 1].

seed (int) - por defecto None
Semilla de número aleatorio.

Half-space trees (HST)

Variante incremental de Isolation Forest

River

anomaly.HalfSpaceTrees

```
import pandas as pd
from river import anomaly
from river import preprocessing

# Cargar el dataset creditcard.csv
file_path = './creditcard.csv'
df = pd.read_csv(file_path)

# Seleccionar las columnas numéricas que serán usadas para el algoritmo
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Inicializar el escalador estándar
scaler = preprocessing.StandardScaler()

# Inicializar el detector de anomalías Half-Space Trees
hst = anomaly.HalfSpaceTrees(n_trees=15)

# Contador de anomalías
anomalies_count = 0

# Ajustar el umbral de anomalía para ser más estricto
anomaly_threshold = 0.98  # Mayor valor --> Menos anomalías
```

Valor a modificar en función de la sensibilidad que se le quiera dar al detector

```
# Iterar sobre cada fila del DataFrame
for i, row in df.iterrows():
    # Diccionario con valores de fila para columnas numéricas
    data_point = {col: row[col] for col in numeric_columns}

    # Actualizar el escalador con el punto actual
    scaler.learn_one(data_point)

    # Estandarizar el punto de datos
    data_point = scaler.transform_one(data_point)

    # El modelo es entrenado al menos una vez antes de calcular la puntuación
    if hst.n_trees > 0:
        # Puntuación de anomalía de HST para el punto actual
        anomaly_score = hst.score_one(data_point)
    else:
        anomaly_score = 0

    # Actualizar el modelo de HST con el punto actual
    hst.learn_one(data_point)

    # Si puntuación de anomalía >umbral --> anomalía
    if anomaly_score > anomaly_threshold:
        anomalies_count += 1
        print(f"Anomalía detectada en fila {i} con puntuación {anomaly_score:.2f}")

# Mostrar el total de anomalías detectadas
print(f"Total de anomalías detectadas: {anomalies_count}")
```

Métricas supervisadas para detección de anomalías

Métricas – Detección de anomalías

En detección de anomalías, el problema suele estar **altamente desbalanceado:** Muy pocas anomalías frente a muchos datos normales.

		Predicted 0	Predicted 1
Actual	0	TN	FP
Actual	1	FN	TP

¿De las anomalías detectadas, cuántas eran reales?

$$Precision = \frac{TP}{TP + FP}$$

Importante cuando el coste de falsa alarma es alto

¿Cuántas anomalías reales detectamos?

$$Recall = \frac{TP}{TP + FN}$$

Fundamental en sistemas críticos (fallos industriales)

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$ Métrica recomendada en anomalías industriales

Evaluación en Streaming

En modo online se pueden usar:

- Prequential evaluation
- Métricas acumulativas
- Ventanas deslizantes de evaluación
- Detección de drift + degradación de F1

Desarrollo de la práctica

64

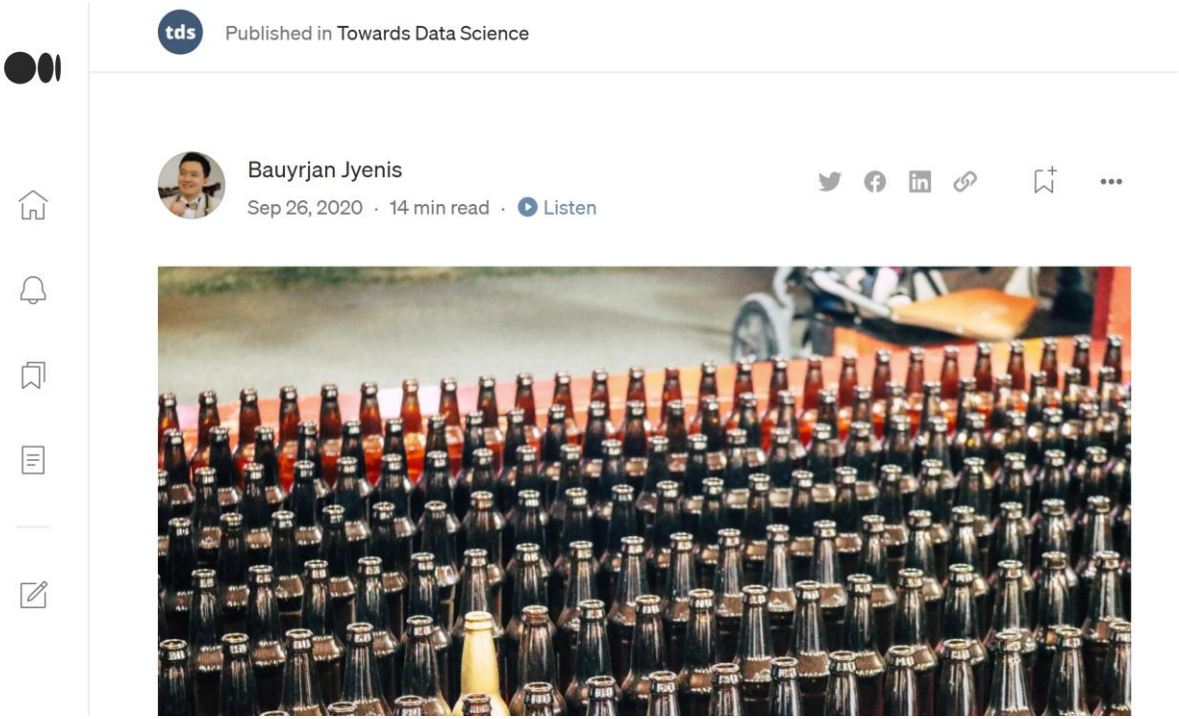
Desarrollo de la práctica

Esta práctica tiene como objetivo desarrollar e implementar técnicas de detección de **anomalías (outliers)** en el funcionamiento de una **bomba de agua equipada con 52 sensores**, los cuales generan una medición cada minuto. El conjunto de datos contiene más de **220.000 registros por sensor**, lo que ofrece un escenario ideal para aplicar métodos de análisis tanto **offline (datos históricos completos)** como **online (streaming o flujo de datos en tiempo real)**, simulando condiciones reales de operación.

Para esta práctica se pide:

- Aplicar y comparar distintas técnicas de detección de anomalías.
- Analizar y contrastar los resultados obtenidos en modo batch (offline) vs. Streaming (online).
- Demostrar comprensión del problema y flexibilidad para adaptar y modificar las técnicas según el contexto.
- Justificar las decisiones técnicas e interpretar los resultados.

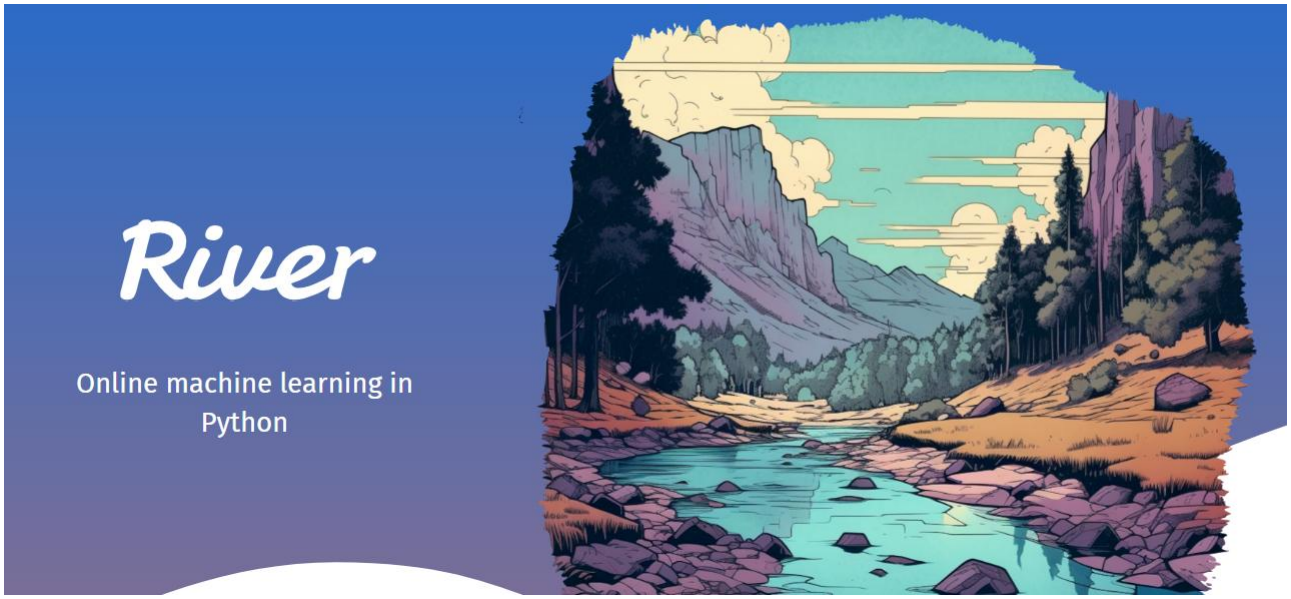
La “parte no incremental” de la práctica puede estar basada en…



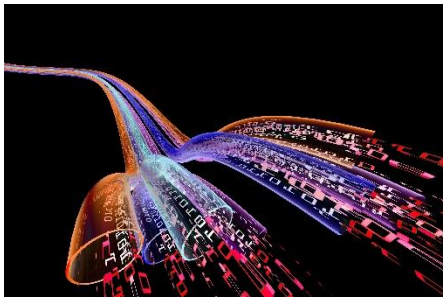
https://towardsdatascience.com/anomaly-detection-in-time-series-sensor-data-86fd52e62538

… pero debe explicarse en detalle (y entenderse) cómo se realiza.

Para los algoritmos incrementales de la práctica debe usarse la librería…



… y explicar bien cuál es la ventaja y utilidad de estos algoritmos que tratan los datos como flujos de datos.



PRÁCTICAS

APLICACIONES AVANZADAS DE LA IA

MÁSTER UNIV. EN INGENIERÍA INFORMÁTICA

Curso: 2024-2025

José Antonio Iglesias Martínez

uc3m