PRÁCTICA 4
IA en la Ingeniería

Aplicaciones Avanzadas de la IA

Máster Universitario en Ingeniería Informática (2024/25)

Detección de anomalías en flujos de datos

Introducción

La prevención de fallos y daños, tal y como se indica en [1], se conoce como el problema de encontrar patrones en los datos que no se ajustan a un comportamiento esperado [2]. Los patrones inesperados suelen denominarse anomalías, valores atípicos o fallos, dependiendo del ámbito de aplicación. En términos generales, las anomalías son patrones en los datos que no se ajustan a un comportamiento normal bien definido. También existen amplios estudios sobre técnicas de detección de anomalías.

En la literatura se han propuesto técnicas de detección de anomalías basadas en la distribución, la distancia, la densidad, agrupación y clasificación. Sus aplicaciones varían según los dominios del problema e incluso del conjunto de datos. En muchos casos, la detección de anomalías está relacionada con la detección de valores atípicos. En estadística los valores atípicos son instancias de datos que se desvían de una muestra determinada en la que se producen. Grubbs en [3] definió una observación atípica, o "outlier", como aquella que parece desviarse notablemente de otros miembros de la muestra en la que se produce.

Las técnicas de detección de anomalías son muy diversas, destacamos a continuación tres de ellas:

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

Cada una de estas técnicas se detalla en el Anexo.

En la industria manufacturera se utilizan varios tipos de maquinaria con motores, bombas, tuberías, hornos, cintas transportadoras, camiones de transporte. A menudo se consideran los activos más críticos para sus operaciones. Por lo tanto, la integridad y la fiabilidad de estos equipos es un objetivo esencial.

La razón principal por la que se preocupan tanto por estos activos es que el fallo de estos equipos a menudo da lugar a pérdidas de producción que, en consecuencia, podrían suponer grandes pérdidas que varían en función del tamaño y la escala de las operaciones. Por lo tanto, es un asunto importante que un director de mantenimiento de una planta de fabricación dirija un marco sólido de gestión de activos con ingenieros de fiabilidad altamente cualificados para garantizar la fiabilidad y la disponibilidad de estos activos críticos.

Objetivo

La capacidad de detectar anomalías con antelación y ser capaz de mitigar los riesgos es una capacidad muy valiosa que permite además prevenir un tiempo de inactividad no planificado, el mantenimiento innecesario y también permitirá una forma más eficaz de gestionar los componentes críticos para estos activos. Las pérdidas de producción derivadas de las paradas no planificadas, el coste del mantenimiento innecesario y el exceso o escasez de componentes críticos se traducen en graves magnitudes en términos de dólares.

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.

Dado que estos datos son públicos, seguro que hay muchos investigadores que han tratado, analizado y obtenido grandes resultados con ellos, por lo que es fundamental que, en la entrega realizada, se vea de forma clara que el alumno ha entendido, comprendido y es capaz de modificar cualquier aspecto relacionado con la resolución del problema planteado.

Descripción de los datos a utilizar

Para esta práctica, se dispondrá del siguiente dataset:
https://www.kaggle.com/nphantawee/pump-sensor-data

Este dataset está formado por las mediciones de 52 sensores de una bomba de agua cada minuto desde el día 1 de abril de 2018 hasta el 31 de Agosto de 2018. Así, cada día se obtienen 1440 mediciones de cada sensor. En total, el conjunto de datos tiene más de 220.319 instancias.

Desarrollo de la práctica

La predicción de anomalías (*outiliers*) se realizará utilizando los 52 sensores para la predicción. Sin embargo, el alumno deberá realizar una reducción de dimensionalidad utilizando técnicas como PCA (tal y como se indica en el documento *DeteccionOutliers.pdf)* de forma que el número de características (componentes principales) puede ser reducido a dos.

Además, para la detección de anomalías (*outliers*), se deberán utilizar 3 algoritmos no-incrementales (Uso completo del Dataset):

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

Y al menos 2 algoritmos incrementales que traten los datos de los sensores como flujos de datos (Uso de mediciones en tiempo real):

- incremental K-Means
- Half Space Trees

Para el uso de algoritmos incrementales se utilizará la librería *River* (https://riverml.xyz/0.22.0/), la cual contine algoritmos de *machine learning* sobre flujos de datos (*streaming data*).

Además, si se desea, se podrán plantear nuevos algoritmos (tanto incrementales como no-incrementales). Es importante que se detallen los resultados de los algoritmos no-incrementales y los incrementales, comprobando si hay diferencias en sus resultados y cuáles serán los más adecuados para el caso planteado.

Por último, este problema ha sido planteado y desarrollado utilizando algoritmos no incrementales en *https://towardsdatascience.com/anomaly-detection-in-time-series-sensor-data-86fd52e62538* (tal y como se propone en el documento *DeteccionOutliers.pdf*). Sin embargo, dado que en el documento adjunto se determinan y codifican una serie de pasos para resolver el problema, es importante que en la entrega de la práctica se observe que se ha entendido, comprendido y se es capaz de modificar cualquier aspecto relacionado con la resolución del problema planteado. También puedes echar un vistazo a: https://becominghuman.ai/predict-pump-failure-before-it-happens-using-deep-learning-model-dc886bfa073e que también trata este dataset para la detección de anomalías utilizando Modelos de Deep Learning.

Por otra parte, para el uso de datos incrementales, es importante consultar el API de River.

Entrega de la práctica

- La práctica deberá realizarse en grupos de **3-4 personas** (y entregarse únicamente por uno de los integrantes del grupo)**.**
- Dicha entrega constará de:
    - Documento explicativo que detalle la propuesta desarrollada, las herramientas utilizadas, los resultados y su análisis, conclusiones, etc…
      No hay un formato de entrega establecido, y se si quiere, el documento podría ser el notebook de Python. Sin embargo, es importante que toda la información se muestre de forma clara y su presentación también es considerada para la nota de la práctica.
    - Aquellos ficheros que sean necesarios para "replicar" el sistema.
- La entrega de esta Práctica deberá realizarse por Aula Global antes del día **11 de abril de 2025 a las 15:00h.**

Evaluación de la práctica

Aspectos que evaluar en la corrección de práctica:

- Planteamiento y desarrollo del problema: 25%
- Resultados del problema: 25%
- Análisis de resultados y conclusiones: 25%
- Presentación: 25%

Bibliografía

[1] L. Martí, N. Sanchez, J. M. Molina y A. C. B. Garcia, «Anomaly Detection Based on Sensor Data in Petroleum Industry Applications,» *Sensors*, vol. 15, nº 2, 2015.

[2] V. Chandola, A. Banerjee y V. Kumar, «Anomaly detection: A survey,» *ACM Comput. Surv. (CSUR)*, vol. 41, 2009.

[3] F. Grubbs, « Procedures for detecting outlying observations in samples,» *Technometrics*, vol. 11, pp. 1-21, 1969.

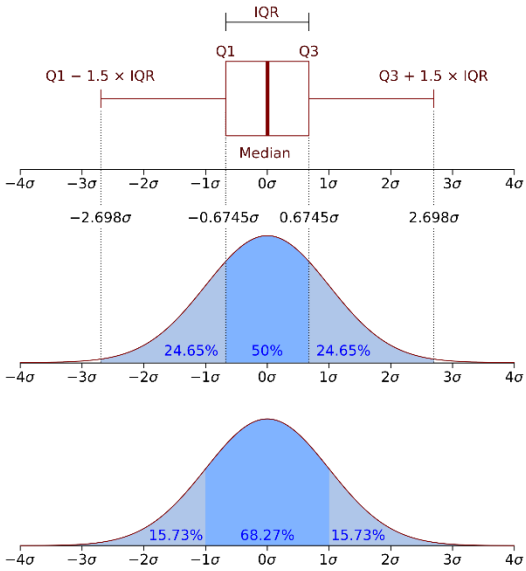
Anexos:

Rango Intercuartil:

En estadística descriptiva, el rango intercuartil (IQR), también llamado dispersión media, 50% medio o dispersión H, es una medida de dispersión estadística, que equivale a la diferencia entre los percentiles 75 y 25, o entre los cuartiles superior e inferior:

IQR = Q3 - Q1.

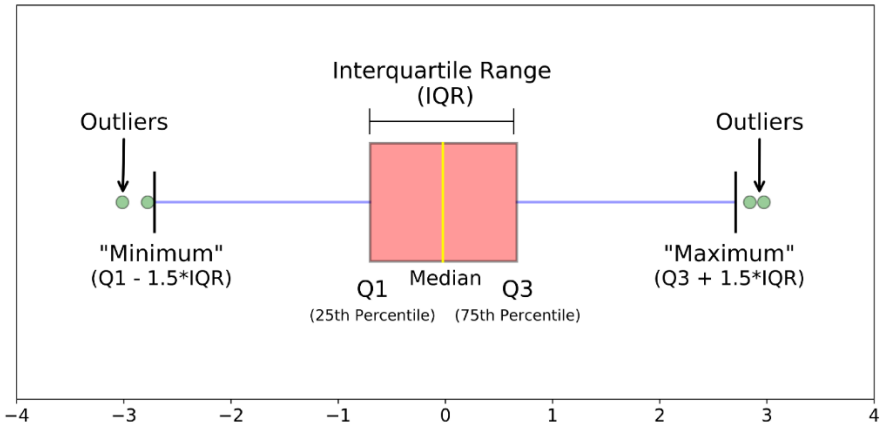
El IQR es una medida de variabilidad, basada en la división de un conjunto de datos en cuartiles. Los cuartiles dividen un conjunto de datos ordenados en cuatro partes iguales. Los valores que separan las partes se denominan primer, segundo y tercer cuartil, y se denominan Q1, Q2 y Q3, respectivamente.



Un uso práctico de la IQR es detectar valores atípicos en los datos. La regla general es que los valores atípicos son observaciones que se encuentran:

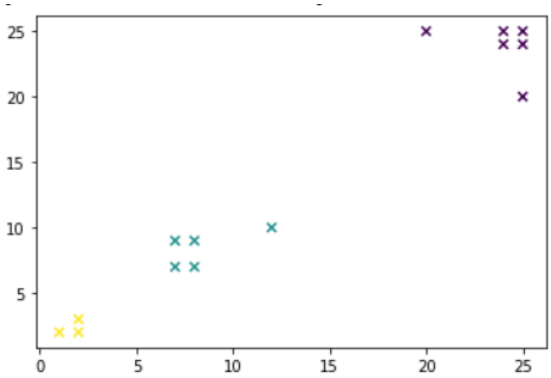
- por debajo del percentil 25 - 1,5 * IQR, o
- por encima del percentil 75 + 1,5 * IQR

De hecho, la siguiente gráfica muestra el gráfico de caja que se crea a partir de los datos:



Referencia: IQR.pdf *(https://medium.com/@prashant.nair2050/hands-on-outlier-detection-and-treatment-in-python-using-1-5-iqr-rule-f9ff1961a414)*

K-Means Clustering:



k-means se utiliza para agrupar los datos en k grupos. Sin embargo, también puede ser utilizado para detectar *"outliers"* considerando aquellos valores que son atípicos en estos grupos. Para ello, es necesario definir un umbral, ya que los datos que permanezcan fuera de la relación de umbral se considerarán como valores atípicos.

Al observar el gráfico, es más fácil ver los puntos que pretendemos detectar como *"outliers"*. En el grupo amarillo no hay ningún valor atípico y hay uno y dos en los grupos verde y morado respectivamente.

Para ello, predecimos los *clusters* de k-means estableciendo un determinado k. Posteriormente, se detectan los centros de los *clusters* y luego calcularemos las distancias entre cada punto y los centros de su correspondiente *cluster.*

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.

Referencia: *KMeans.pdf (https://medium.datadriveninvestor.com/outlier-detection-with-k-means-clustering-in-python-ee3ac1826fb0)*

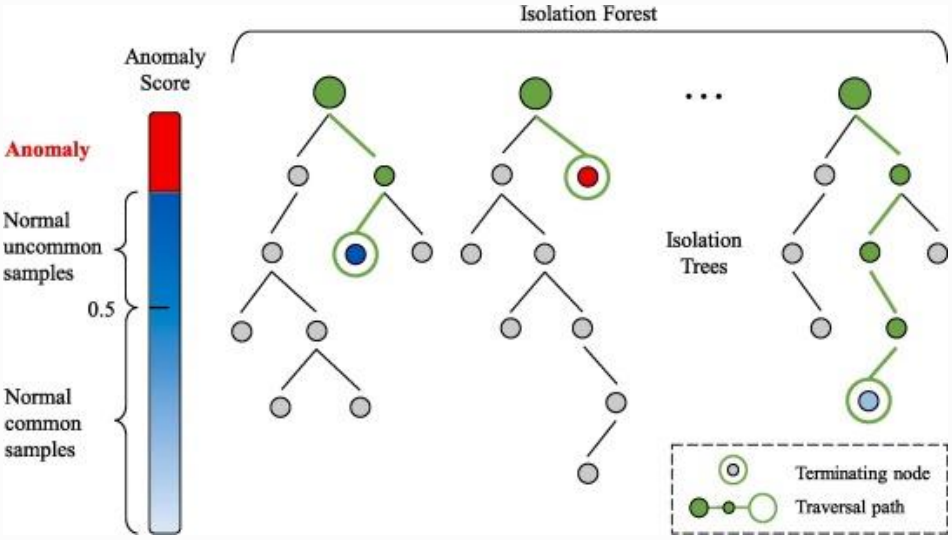
Em *DeteccionOutliers.pdf* (*https://towardsdatascience.com/anomaly-detection-in-time-series-sensor-data-86fd52e62538*) se proponen los siguientes pasos:

1. Calcular la distancia entre cada punto y su centroide más cercano. Las distancias más grandes se consideran anomalías.
2. Utilizar *outliers_fraction* para proporcionar información al algoritmo sobre la proporción de los valores atípicos presentes en nuestro conjunto de datos. La situación puede variar de un conjunto de datos a otro y dicho valor debe ser considerado.
3. Calcular el número de valores atípicos utilizando la fracción de valores atípicos del punto anterior.
4. Establecer el umbral como la distancia mínima de estos valores atípicos.

Isolation Forest:

Isolation Forest es un 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*, un modelo *Isolation Forest* está formado por la combinación de múltiples árboles llamados *isolation trees*. Estos árboles se crean de forma similar a los de clasificación-regresión: las observaciones de entrenamiento se van separando de forma recursiva creando las ramas del árbol hasta que cada observación queda aislada en un nodo terminal. 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.



Algoritmo Isolation Tree

1. Crear un nodo raíz que contiene las N observaciones de entrenamiento.
2. Seleccionar aleatoriamente un atributo i y un valor aleatorio dentro del rango observado de i
3. Crear dos nuevos nodos separando las observaciones acorde al criterio:
   $x_i \leq a$ o $x_i > a$
4. Repetir los pasos 2 y 3 hasta que todas las observaciones quedan aisladas de forma individual en nodos terminales.

El modelo **Isolation Forest** se obtiene al combinar múltiples *isolation tree*, cada uno entrenado con una muestra distinta generada por *bootstrapping* a partir de los datos de originales. El valor predicho para cada observación es el número de divisiones promedio que se han necesitado para aislar dicha observación en el conjunto de árboles. Cuanto menor es este valor, mayor es la probabilidad de que se trate de una anomalía.

Consideraciones prácticas

Al ser un método no supervisado, no hay forma de conocer el valor óptimo a partir del cual se debe de considerar que se trata de una anomalía. La puntuación asignada a cada observación es una medida relativa respecto al resto de observaciones. En la práctica, suelen considerarse como potenciales *outliers* aquellas observaciones cuya distancia predicha está por debajo de un determinado cuantil.

Cuando se dispone de muchas observaciones, aislarlas todas en nodos terminales requiere de árboles con muchas ramificaciones, lo que se traduce en un coste computacional muy elevado. Una forma de aliviar este problema es determinar una profundidad máxima hasta la que se puede crecer el árbol. A aquellas observaciones que, una vez alcanzado el criterio de parada, no han llegado a nodos terminales individuales, se les suma el número de divisiones teóricas promedio c(r)c(r) que se necesita para aislar mediante particiones binarias un nodo de rr observaciones.

$$c(r)=\log(r-1)-2(r-1)r+0.5772$$

Referencia: *https://rpubs.com/Joaquin_AR/614976*

K-Means Incremental:

https://riverml.xyz/0.15.0/api/cluster/KMeans/

El método de clustering k-means en mini lotes (Mini-batch K-Means) o K-Means Incremental es una versión de **aprendizaje incremental del algoritmo de clustering k-means** que se puede aplicar directamente a un flujo de datos.
Se trata de una variante del algoritmo k-means que funciona con conjunto pequeño de datos (mini-lotes) en lugar de en todo el conjunto de datos.
Es adecuado para el flujo de datos, ya que un conjunto de datos completo nunca está disponible en ningún momento. Los minilotes son subconjuntos del flujo de datos de entrada. Se muestrean aleatoriamente para cada iteración de entrenamiento.

Para aplicarse con un flujo de datos:

- Se recibe una 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.

Parámetros relevantes:

- **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.

Ejemplo en AG: ***IncrementalKMeans.ipynb***

Half-Space Trees (HST):

https://riverml.xyz/0.15.0/api/anomaly/HalfSpaceTrees/

Half-Space Trees es la variante incremental de Isoltaion Forest. Esta variante funciona bien cuando las anomalías son repetidas. Sin embargo, si las anomalías están agrupadas en ventanas, no funciona tan bien.

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.

Parámetros relevantes:

- **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 la masa en 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.

Ejemplo en AG: ***HalfSpaceTrees.ipynb***