Lección 5. Comparando métodos de detección de anomalías

UCO ONLINE

La detección de anomalía

Comparando métodos de detección de anomalías

En este curso se han visto diferentes técnicas de detección de anomalías. En función del tipo de datos, así como de sus características, determinadas técnicas pueden resultar más o menos adecuadas para resolver el problema.

En este tema veremos el proceso de realizar un pequeño estudio experimental que nos permita comparar diferentes técnicas para resolver un determinado problema. Nos centraremos primero en el procesamiento de los datos, eligiendo uno de los conjuntos de datos y su procesamiento. Posteriormente, se describirá un estudio comparativo de diferentes técnicas usando las métricas estudiadas.

1. Introducción

Los pasos que vamos a seguir para llevar a cabo el estudio experimental se indican en el diagrama 1. El primer paso (fase 1) es seleccionar un conjunto de datos, utilizando un método que nos permita leerlo desde Python, habrá que dividirlo en training y test para poder realizar el experimento. Además, para los métodos que utilizamos sería conveniente utilizar un método de normalización de los datos. En el segundo paso (fase 2), configuraremos los algoritmos que se van a utilizar, diferentes valores en los parámetros pueden afectar a los resultados que se obtengan. Como se ha visto con los ejemplos anteriores durante estas semanas, cambiando los parámetros se podía optimizar los resultados. El siguiente paso sería ejecutar los algoritmos y obtener las métricas que se están estudiando (fase 3), para finalmente, pasar a analizarlas (fase 4).

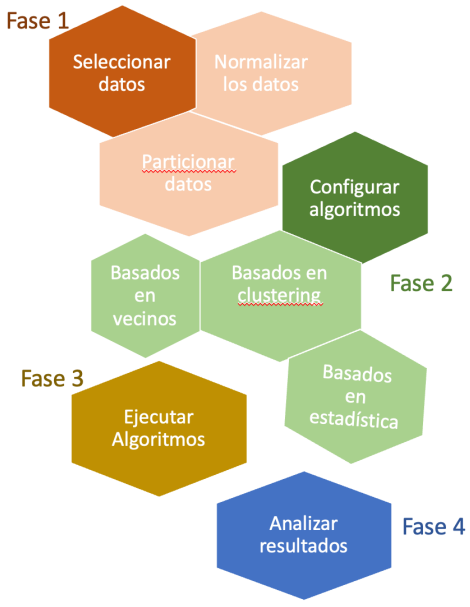

Diagrama 1. Pasos a seguir en un estudio experimental para comparar algoritmos

2. Seleccionar conjunto de datos (Fase 1)

El primer paso es seleccionar el problema que se va a resolver. Para ello, debemos partir de un conjunto de datos que tenga ejemplos del problema tanto con anomalías como sin anomalías. Estos datos serán utilizados por los diferentes métodos para aprender y ser capaces de detectar las anomalías. Existen varios conjuntos de datos con los que podemos trabajar, nosotros trabajaremos con datos que se encuentran en la web ODDS (*Outlier Detection DataSets*):

http://odds.cs.stonybrook.edu/

Dicha web tiene accesible una gran colección de conjuntos de datos de problemas reales para detección de anomalías. Los conjuntos de datos se clasificación en diferentes tipos bajo un mismo formato:

- Conjuntos de datos de puntos multivariantes: hay un registro por punto de datos y cada registro contiene varios atributos.
- Conjuntos de datos de gráficos de series temporales para la detección de eventos: datos de series temporales en los que el gráfico cambia dinámicamente al tiempo que llegan nuevos eventos o desaparecen eventos.
- Conjuntos de datos de puntos de series temporales (multivariante/univariante): datos de puntos temporales donde cada punto tiene uno o más atributos y los atributos cambian con el tiempo.
- Conjuntos de datos de escenarios adversos/de ataque y de seguridad: datos de detección de fraude. Datos de ciberseguridad, detección de intrusos con escenario de ataque DoS, DDoS, etc.
- Datos de vídeo de escenas para la detección de anomalías: clips de vídeo adquiridos con la cámara.

Estos conjuntos de datos reciben normalmente el nombre en inglés de *dataset* y se trata de una representación de datos en forma de una única tabla bidimensional, siendo cada columna una variable o atributo (por ejemplo: color y altura) y cada fila representa un ejemplo o instancia del conjunto de datos (por ejemplo: amarillo para el atributo color o 1.80 para el atributo altura). La unión de todas las filas y columnas proporciona el conjunto de todos los valores que pueden tener las variables. El tipo de datos de cada atributo puede ser numérico o categórico.

La investigación de detección de anomalías se ha visto frenada por falta de buenos conjuntos de datos con buenas referencias. Actualmente muchos problemas utilizados no están disponibles o son artificiales.

2.1 Información de los conjuntos de datos seleccionados

En nuestro estudio nos basaremos en los elementos de la primera categoría (conjuntos de datos de puntos multivariantes). Es decir, nos centraremos en las anomalías puntuales que son representadas por puntos/ejemplos en el espacio de acuerdo con las dimensiones establecidas por cada conjunto de datos. Este tipo de conjuntos está compuesto por un registro por cada punto de datos, y cada registro contiene varios atributos.

En la tabla 1 se muestra información de los tres conjuntos de datos que se han seleccionado para este ejemplo. En esta tabla se indica: el nombre del conjunto de datos (#dataset), el número de ejemplos/instancias que tienen (#ejemplos), cuántos atributos tiene para representar cada instancia (#dimensiones) y el porcentaje de anomalías que tienen (#anomalías).

#Dataset	**#Ejemplos**	**#Dimensiones**	**#Anomalías(%)**
Arrhythmia	452	274	66 (15%)
Cardio	1831	21	176 (9.6%)
Glass	214	9	9 (4.2%)

Tabla 1. Conjunto de datos para detección de anomalías

2.1.1 Conjunto de datos - Arrhythmia

El conjunto de datos original de arrhythmia es del repositorio de la UCI. Los datos corresponden con un estudio de H. Altay Guvenir donde el objetivo fue distinguir entre la presencia y ausencia de arritmia cardiaca y clasificarla en uno de los 16 grupos. La clase 01 se refiere a ECG 'normal'. Las clases 02 a 15 se refieren a diferentes clases de arritmia y la clase 16 se refiere al resto de las no clasificadas. Este es un conjunto de datos de clasificación con lo que tenemos información de las clases de cada ejemplo y podemos evaluar nuestros métodos una vez tengan un resultado. Las clases más pequeñas se unen para formar las anomalías y el resto se une para formar la clase normal.

El conjunto de datos cuenta con 452 instancias y 274 dimensiones de las que contiene 66 anomalías que representan un 15% del conjunto de datos.

2.1.2 Conjunto de datos - Cardio

El conjunto de datos original de cardio es del repositorio de la UCI y consta de mediciones de la frecuencia cardíaca fetal (FHC) y las características de contracción uterina (UC) en cardiotocogramas clasificados por obstetras expertos. Este es un conjunto de datos de clasificación donde las clases son normales, sospechosas y patológicas. Para la detección de valores atípicos, la clase normal formó los valores normales, mientras que la clase patológica (valores atípicos) se reduce a 176 puntos y la clase sospechosa se descarta.

El conjunto de datos cuenta con 1831 instancias y 21 dimensiones de las que contiene 176 anomalías que representan un 9.7% del conjunto de datos.

2.1.3 Conjunto de datos - Glass

El conjunto de datos original de glass es del repositorio de la UCI. El conjunto de datos trata la identificación del vidrio original. El estudio de la clasificación de los tipos de vidrio estuvo motivado por la investigación en criminología. En la escena del crimen, el vidrio dejado puede usarse como evidencia, si se identifica correctamente. Este conjunto de datos contiene atributos relacionados con varios tipos de vidrio (múltiples clases). Este es un conjunto de datos de clasificación, donde la clase minoritaria (clase 6) se marca como valores atípicos, mientras que todos los demás puntos son normales.

El conjunto de datos cuenta con 214 instancias y 9 dimensiones de las que contiene 9 anomalías que representan un 4.2% del conjunto de datos.

2.2 Preparación del conjunto de datos

A la hora de aplicar cualquiera de los modelos que hemos estudiado es de gran importancia el uso de los conjuntos de datos que se utilizan para entrenar un modelo. A la hora de preparar los datos vamos a ver el proceso de particionado y el de normalización.

2.2.1 División entre conjunto de prueba y entrenamiento

Como se ha visto en los ejemplos donde se han analizado y aplicado los modelos, resulta necesario partir de un conjunto de datos de entrenamiento y uno de test. El conjunto de entrenamiento es el subconjunto de datos que vamos a usar para entrenar el modelo. Este subconjunto se utilizará para que los algoritmos puedan aprender a resolver el problema. El conjunto de test es el subconjunto que vamos a usar para probar el modelo entrenado utilizando unos datos que no ha visto antes y simulando que serían los nuevos datos que le llegaría y queremos que identifique si se trata de una anomalía. Por tanto, este subconjunto de datos se utiliza para proporcionar una evaluación y análisis de los resultados imparcial del modelo que se ha entrenado en el paso anterior.

Es de gran importancia que el conjunto de prueba reúna las siguientes tres condiciones:

- Debe ser lo suficientemente grande para generar resultados significativos desde el punto de vista estadístico.
- Debe ser representativo de todo el conjunto de datos. La elección del conjunto de prueba no debe tener características diferentes al del conjunto de entrenamiento. De esta forma, en el problema que nos ocupa, la proporción de anomalías debe ser similar en entrenamiento y test.
- Debe mantenerse separado de los datos de entrenamiento. Las observaciones asignadas al subconjunto de test no deben ser introducidas en los cálculos con el subconjunto de entrenamiento.

A la hora de realizar el particionado existen diferentes propuestas:

- **Método de retención (*holdout*):** este método se utiliza cuando la cantidad de datos para entrenamiento y prueba es limitada. Consiste en reservar un porcentaje de datos para la prueba y el resto para entrenamiento, se suele considerar un tercio de los datos para la prueba y los otros dos tercios restantes para entrenamiento [1].

  El principal inconveniente que presenta este método es que puede existir una variación en la estimación del error y no resultar representativos los datos de entrenamiento y prueba, los cuales suelen ser clasificados en cada conjunto de forma aleatoria.

- **Método de validación cruzada (*cross validation*):** es un método de evaluación de modelos ampliamente utilizados, tiene como objetivo hacer un uso efectivo de todas las instancias del conjunto de datos tanto para entrenamiento como para test [2]. Este método permite utilizar todas las observaciones disponibles para entrenamiento y al mismo tiempo usar muchas observaciones como conjunto de pruebas independientes.

  Se puede diferenciar dos tipos principalmente:

- **Validación cruzada dejando uno afuera (Leave-one-out cross-validation (LOOCV)):** en este caso todos los datos se usan para entrenamiento y son utilizados por el método de aprendizaje, excepto uno que se usa para test. Cuando se usa una observación para test, esa no se utiliza en el entrenamiento. Esto se repite para las n instancias, luego se promedian los resultados y ese promedio representa la estimación de error final. El error que se estudia con esta validación es muy bajo ya que se evalúa el rendimiento del algoritmo para cada observación de forma independiente.

  Este procedimiento permite utilizar la mayor cantidad de observaciones para el entrenamiento, lo que a priori aumenta la probabilidad de que el modelo sea exacto. Al no usar muestras al azar, no depende de la distribución de las observaciones. Se puede obtener una estimación lo más precisa posible para pequeños conjuntos de datos. Sin embargo, tiene alto costo computacional, por lo que no es factible para grandes conjuntos de datos. El problema es que computacionalmente es muy costoso y por tanto, inviable de aplicar, ya que se realiza *n* iteraciones, donde n es el número de instancias que tiene el conjunto de datos.

- **Validación cruzada de k iteraciones (k-fold cross validation)**: consiste en dividir los datos originales en *k* subconjuntos y, al momento de realizar el entrenamiento, se va a tomar cada *k* subconjunto como conjunto de test del modelo, mientras que el resto de subconjuntos (*k*-1) se tomará como conjunto de entrenamiento. El proceso de validación cruzada se repetirá *k* veces, y en cada iteración se tomará un conjunto de test diferente, siendo el resto de datos el conjunto de entrenamiento. Al finalizar las k iteraciones, se calcula el promedio de los resultados de precisión y error obtenidos para cada subconjunto de prueba [2].

En la validación cruzada *k-fold*, los datos iniciales se dividen aleatoriamente en *k* subconjuntos mutuamente exclusivos, D_1, D_2, ..., D_k, cada uno aproximadamente del mismo tamaño. El entrenamiento y las pruebas se realizan *k* veces. En general, se recomienda una validación cruzada estratificada de 10-fold para estimar la precisión, debido a su sesgo y varianza relativamente bajos [2].

En nuestro caso y para facilitar el tiempo y proceso de la generación de resultados vamos a trabajar con el método de holdout. En cuanto a la selección del porcentaje, aunque no hay una regla empírica que resuelva el problema, se acepta por la comunidad que, con menos datos de entrenamiento, las estimaciones de sus parámetros tienen mayor variabilidad y que con menos datos de test, su estadística de rendimiento tendrá mayor varianza. En la mayoría de los casos la selección suele atribuir un porcentaje entre 20%-30% al conjunto de prueba. En nuestro caso tomaremos el 30%, dejando el 70% del conjunto de datos en el subconjunto de entrenamiento.

Con respecto a estos resultados de entrenamiento y test, es importante introducir dos conceptos sobreajuste (*overfitting)* y subajuste (*underfitting*) [3].

- **El subajuste** se produce cuando un modelo estadístico o un algoritmo de aprendizaje automático obtiene un bajo rendimiento debido a que no tenía suficientes datos para poder captar la tendencia subyacente de los mismos. El modelo probablemente realizará un gran número de predicciones erróneas. Las causas posibles de que esto ocurra son:
  - Modelo demasiado simple, no es capaz de aprender relación entre datos de entrada y salida.
  - Falta de datos: si no contamos con los datos necesarios el rendimiento podría ser malo. Por ejemplo, nos falte alguna de las variables que se deben evaluar.
  - Atributos relevantes insuficientes Es posible tener todos los datos pero que surja la necesidad de transformarlos para una mejor comprensión del modelo entre la relación de entrada salida.

  Lo identificamos porque los resultados con los datos de entrenamiento no son buenos.

- **El sobreajuste** se produce cuando un modelo estadístico o un algoritmo de aprendizaje automático obtiene un bajo rendimiento debido a que se entrena con un número demasiado alto de datos. Cuando un modelo se entrena con datos de más, comienza a aprender también el ruido y las entradas de datos inexactas y no es capaz de categorizar los datos correctamente debido a demasiados detalles. Las causas posibles de que esto ocurra son:

- Modelo demasiado complejo, podrá aprender muchos de los datos de memoria.
- Los datos tienen ruido y errores en los datos.
- El tamaño de los datos utilizados para los datos de entrenamiento o training data puede que no sean suficientes.

Se detecta porque los resultados del entrenamiento son perfectos y en el test no salen bien.

Idealmente, el caso en que el modelo hace las predicciones con error 0, se dice que tiene un buen ajuste en los datos. Esta situación es alcanzable en un punto entre el sobreajuste y el subajuste. Si un modelo aprende durante demasiado tiempo, el modelo será más propenso a la sobrecarga debido a la presencia de ruido y de detalles menos útiles. Por lo tanto, el rendimiento de nuestro modelo disminuirá. Para conseguir un buen ajuste, nos detendremos en un punto justo antes de que el error comience a aumentar. En este punto se dice que el modelo tiene buenas habilidades en los conjuntos de datos de entrenamiento, así como en nuestro conjunto de datos de pruebas no vistas.

Para realizar este paso, tienes disponible el método train_test_split de la librería sklearn, donde obtendremos los datos de entrenamiento X_train, los datos de test X_test y las salidas de si son o no anomalías y_train e y_test.

```
# Divimos los datos en train y test (70%, 30% respectivamente). También
# se pueden usar otras formas de cross-validation
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=random_state)
```

2.2.2 Normalización de los datos

La normalización de los conjuntos de datos es un requisito común de muchos modelos de aprendizaje automático. Estos modelos podrían comportarse mal si las características individuales no se parecen a los datos estándar distribuidos normalmente [4] con media cero y varianza unitaria.

El objetivo de la normalización en nuestro caso va a consistir en cambiar los valores de las columnas numéricas del conjunto de datos (atributos variable X) a una escala común, sin distorsionar las diferencias en los rangos de valores. Para el aprendizaje automático, no es necesario normalizar todos los conjuntos de datos. Sólo se requiere cuando las características tienen rangos diferentes. Si un atributo tiene una varianza de órdenes de magnitud mayores que otras, podría dominar las medidas de distancia y hacer que el estimador no pueda aprender de otras características correctamente como se espera.

En la práctica, ignoraremos la forma de la distribución y simplemente transformamos los datos para centrarlos eliminando el valor medio de cada característica, y luego los escalaremos dividiendo las características no constantes por su desviación típica. Para realizar este paso, tienes disponible el método standardizer de la librería pyod, donde se lleva a cabo la normalización Z de los datos para que las muestras de entrada tengan media cero y varianza unitaria. Obtenemos los datos normalizados en X_train_norm y X_test_norm

```
# Normalizamos los datos (usamos el métodos de PyOD)
X_train_norm, X_test_norm = standardizer(X_train, X_test)
```

3. Configurar parámetros de los algoritmos (Fase 2)

Una vez tenemos los datos particionados y normalizados, pasamos a la fase 2. Para la correcta evaluación de los algoritmos se realizan diversas pruebas en los distintos algoritmos y sus parámetros. Cuando hemos analizado los diferentes métodos en las semanas anteriores, hemos visto que tenían una serie de parámetros que podían cambiarse y en función de ellos, los resultados logrados por el algoritmo podían mejorarse.

La selección del valor óptimo para los parámetros se realiza estudiando la influencia en diversas ejecuciones de los distintos valores en los resultados finales. En nuestro estudio, vamos a realizarla en base a su influencia en las métricas de sensibilidad y especificidad. Así, haremos una aproximación, realizaremos varias ejecuciones y seleccionaremos el valor con mejor resultado en las diferentes métricas.

La asignación de valores al parámetro se realiza mediante un bucle donde para cada algoritmo y conjunto de datos se realizan diferentes ejecuciones cambiando sus parámetros y se selecciona manualmente el valor que proporciona los mejores resultados y es la configuración que se usará en el estudio final. De los diferentes algoritmos solamente se va a estudiar un parámetro:

- Algoritmos basados en vecinos: parámetro *k*, que indica el número de vecinos a seleccionar para realizar los cálculos. Los algoritmos KNN, LOF y COF.
- Algoritmos basados en agrupamiento: parámetro *n_clusters*, el número de clústeres que indica el número de grupos que se formarán para detectar las áreas densas y seleccionar los datos que queden fuera de ellas marcándolos como anomalías. Algoritmo CBLOF.
- Algoritmos basados en histogramas: parámetro *n_bins*, número de barras que forman el histograma. Algoritmo HBOS.

Para poder llevar a cabo este paso, tienes disponible un notebook para cada uno de los algoritmos, donde solamente se debe seleccionar el conjunto de datos y el rango de valores a analizar para obtener los resultados:

- ***Configurar_algoritmo_KNN.ipynb***: tiene el código disponible para llevar a cabo la configuración del algoritmo KNN de acuerdo al número de vecinos.
- ***Configurar_algoritmo_LOF.ipynb***: tiene el código disponible para llevar a cabo la configuración del algoritmo LOF de acuerdo al número de vecinos.
- ***Configurar_algoritmo_COF.ipynb***: tiene el código disponible para llevar a cabo la configuración del algoritmo COF de acuerdo al número de vecinos.
- ***Configurar_algoritmo_CBLOF.ipynb***: tiene el código disponible para llevar a cabo la configuración del algoritmo CBLOF de acuerdo al número de clústeres.
- ***Configurar_algoritmo_HBOS.ipynb***: tiene el código disponible para llevar a cabo la configuración del algoritmo HBOS de acuerdo al número de barras.

En el vídeo se explican las variables a modificar para poder llevar a cabo este estudio, que sería similar en los diferentes ficheros referentes a cada uno de los algoritmos.

Por ejemplo, usando el fichero *Configurar_algoritmo_LOF* y el conjunto de datos arrhythmia variando el valor de k de 10 a 100 en variaciones de 10, obtenemos la siguiente tabla:

	Algoritmo	Datos	#Ejemplos	#Dimensiones	Anomalías(%)	k	Se	Sp
0	LOF	arrhythmia	452	274	14.6018	10	0.381	0.8375
0	LOF	arrhythmia	452	274	14.6018	20	0.4286	0.8375
0	LOF	arrhythmia	452	274	14.6018	30	0.4762	0.8438
0	LOF	arrhythmia	452	274	14.6018	40	0.4762	0.8375
0	LOF	arrhythmia	452	274	14.6018	50	0.4762	0.8438
0	LOF	arrhythmia	452	274	14.6018	60	0.4762	0.8375
0	LOF	arrhythmia	452	274	14.6018	70	0.4762	0.8438
0	LOF	arrhythmia	452	274	14.6018	80	0.4762	0.8438
0	LOF	arrhythmia	452	274	14.6018	90	0.4762	0.8375

Se puede ver que para *k* igual a 30, 50, 70, 80 se consigue los mejores valores de Se y Sp. Por tanto, el valor que seleccionaríamos es 30, ya que es más reducido y de unos valores a otros siempre varían los resultados. Normalmente, este estudio se lleva a cabo realizando un análisis de sensibilidad de los diferentes parámetros, pero en nuestro caso, realizaremos este breve estudio para poder garantizar una optimización en los resultados del algoritmo.

4. Ejecutar algoritmos sobre el conjunto de prueba (Fase 3)

El siguiente paso sería ejecutar los algoritmos sobre el conjunto de prueba con los parámetros configurados del paso anterior para un mejor rendimiento. En esta fase se ejecuta el algoritmo mediante los datos de entrenamiento para el conjunto de prueba obteniendo como resultados las mismas métricas mostradas en el conjunto de entrenamiento. La ejecución se realiza mediante los valores de los parámetros obtenidos en la fase 2 y se obtienen los resultados del test para analizarlos.

Para poder llevar a cabo este paso, tienes disponible un notebook "*Configurar_modelos_anomalias.ipynb*" que lleva a cabo la comparación de los 5 métodos indicados en el paso anterior para tres conjuntos de datos obteniendo las métricas de sensibilidad, especificidad, precisión y ROC-AUC. Además, del tiempo de cómputo que para determinadas aplicaciones puede ser un factor determinante a la hora de su aplicación.

A continuación, se muestra para cada uno de los conjuntos de datos considerados y con el algoritmo optimizado, las siguientes tablas para cada una de las métricas comentadas:

Tiempo de cómputo

	Datos	#Ejemplos	#Dimensiones	Anomalías(%)	HBOS	CBLOF	KNN	LOF	COF
0	arrhythmia	452	274	14.6018	3.8149	2.2244	0.0307	0.0107	0.1497
0	cardio	1831	21	9.6122	0.2169	0.0142	0.111	0.1137	4.4199
0	glass	214	9	4.2056	0.0896	0.0036	0.006	0.0024	0.0142

Sensibilidad

	Datos	#Ejemplos	#Dimensiones	Anomalías(%)	HBOS	CBLOF	KNN	LOF	COF
0	arrhythmia	452	274	14.6018	0.45	0.6	0.45	0.4	0.4
0	cardio	1831	21	9.6122	0.3617	0.5957	0.4681	0.5745	0.2979
0	glass	214	9	4.2056	1.0	0.5	0.5	1.0	1.0

Especificidad

	Datos	#Ejemplos	#Dimensiones	Anomalías(%)	HBOS	CBLOF	KNN	LOF	COF
0	arrhythmia	452	274	14.6018	0.9397	0.9138	0.931	0.9224	0.9138
0	cardio	1831	21	9.6122	0.9523	0.9543	0.9563	0.9662	0.9563
0	glass	214	9	4.2056	0.9048	0.9206	0.9841	0.9524	0.9524

Precisión

	Datos	#Ejemplos	#Dimensiones	Anomalías(%)	HBOS	CBLOF	KNN	LOF	COF
0	arrhythmia	452	274	14.6018	0.5625	0.5455	0.5294	0.4706	0.4444
0	cardio	1831	21	9.6122	0.4146	0.549	0.5	0.6136	0.3889
0	glass	214	9	4.2056	0.25	0.1667	0.5	0.4	0.4

ROC

	Datos	#Ejemplos	#Dimensiones	Anomalías(%)	HBOS	CBLOF	KNN	LOF	COF
0	arrhythmia	452	274	14.6018	0.6948	0.7569	0.6905	0.6612	0.6569
0	cardio	1831	21	9.6122	0.657	0.775	0.7122	0.7703	0.6271
0	glass	214	9	4.2056	0.9524	0.7103	0.7421	0.9762	0.9762

Realmente, como ya se ha comentado en otras ocasiones, debería ejecutarse varias veces cambiando la semilla aleatoria para obtener los valores medios de los algoritmos que son estocásticos y dependen de esa generación de valores. No obstante, con la idea de simplificar el estudio, solamente tenéis que evaluar una única ejecución.

5. Análisis de los resultados (Fase 4)

La última fase sería analizar los resultados. En este caso, solamente se va a pedir que comparéis dos algoritmos, con lo que la idea sería comentar cuál de ellos obtiene mejores resultados en cada una de las métricas. En esta fase se suelen aplicar test estadísticos que permitan comparar tanto los algoritmos de dos en dos, como varias propuestas simultáneamente [5]. No obstante, con la idea de simplificar el estudio que vais a realizar, no se llevará a cabo ningún test estadístico y simplemente analizaremos los resultados en cada uno de los conjuntos de datos para tomar una conclusión.

Ejemplo de análisis de los resultados para los tres conjuntos de datos: *arrhythmia, cardio y glass*, considerando los algoritmos LOF y COF.

Sensibilidad

	Datos	#Ejemplos	#Dimensiones	Anomalías(%)	LOF	COF
0	arrhythmia	452	274	14.6018	0.4	0.4
0	cardio	1831	21	9.6122	0.5745	0.2979
0	glass	214	9	4.2056	1.0	1.0

Especificidad

	Datos	#Ejemplos	#Dimensiones	Anomalías(%)	LOF	COF
0	arrhythmia	452	274	14.6018	0.9224	0.9138
0	cardio	1831	21	9.6122	0.9662	0.9563
0	glass	214	9	4.2056	0.9524	0.9524

Se puede ver que LOF obtiene mejores valores de sensibilidad y especificada en todos los conjuntos de datos. No obstante, los valores de sensibilidad en el caso de los conjuntos de datos *arrhythmia* y *cardio* se encuentran cercanos al 50%, por lo que no es muy elevado el acierto en la clase de anomalías. Los valores de especificidad que nos determinan el número de aciertos de la clase normal, podemos ver que se encuentran en valores superiores al 90%.

Precisión

	Datos	#Ejemplos	#Dimensiones	Anomalías(%)	LOF	COF
0	arrhythmia	452	274	14.6018	0.4706	0.4444
0	cardio	1831	21	9.6122	0.6136	0.3889
0	glass	214	9	4.2056	0.4	0.4

ROC

	Datos	#Ejemplos	#Dimensiones	Anomalías(%)	LOF	COF
0	arrhythmia	452	274	14.6018	0.6612	0.6569
0	cardio	1831	21	9.6122	0.7703	0.6271
0	glass	214	9	4.2056	0.9762	0.9762

Si valoramos las medidas de precisión y AUC-ROC, de nuevo, LOF obtiene mejores resultados en todos los conjuntos de datos. En el caso de *glass*, podemos ver que el valor de precisión es de 0.4, este conjunto de datos tiene muy pocas instancias y muy pocos ejemplos anómalos, lo que hace que un fallo afecte considerablemente a los resultados finales. Se puede ver que si analizamos los valores de auc roc para dicho conjunto de datos son muy elevados considerando toda la clasificación en general.

Tiempo de cómputo

	Datos	#Ejemplos	#Dimensiones	Anomalías(%)	LOF	COF
0	arrhythmia	452	274	14.6018	0.0165	0.2898
0	cardio	1831	21	9.6122	0.1438	5.0638
0	glass	214	9	4.2056	0.0038	0.0175

Finalmente, el tiempo de cómputo también es más reducido para LOF. Con lo que podemos concluir, que LOF sería una mejor propuesta frente a COF para resolver estos problemas.

Referencias

[1] I. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques, second edition, San Francisco, CA, EEUU, Morgan Kaufmann, 2005.

[2] T. T. Wong. Performance evaluation of classification algorithms by k- fold and leave-one-out cross validation, Pattern Recognition, 48 (9), 2839-2846, 2015.

[3] A. Bhande. What is underfitting and Overfitting in Machine Learning and how to deal with it [Internet]. [visitado 20 marzo 2022]. Disponible en: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

[4] S. García, J. Luengo, & F. Herrera, F. Data preprocessing in data mining (Vol. 72, pp. 59-139). Cham, Switzerland: Springer International Publishing, 2015.

[5] J. Demšar. Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research, 7, 1-30, 2006.