ISOLATION FOREST

Índice

Introducción

En este documento se explican algunos algoritmos implementados de Isolation Forest, como obtener sus resultados y cómo funcionan.

Instalación

El repositorio utilizado se encuentra en la siguiente url:
https://github.com/Dennis-mon/IAwithTensorFlow

Para su instalación debemos seguir los siguientes pasos:

1. ***Clonar el repositorio:***

```
git clone https://github.com/Dennis-mon/IAwithTensorFlow.git
```

2. ***Movernos a la carpeta isolation_forest***

```
cd ./isolation_forest
```

3. ***Instalar dependencias***

```
npm i
```

4. ***Probar algoritmos. Se pueden hacer pruebas individuales o grupales:***
    a. *individual:*
        i.
```
npm run algoritmo1
```
        ii.
```
npm run algoritmo2
```
        iii.
```
npm run algoritmo3
```
    b. *grupal:*
        i.
```
npm run algoritmos
```

Librerías

- **canvas:** la utilizamos para crear el lienzo donde se colocará el gráfico
- **chart.js:** librería usada para generar distintos tipos de gráficos
- **chartjs-adapter-moment:** sirve para poder ordenar los ejes de los gráficos con fechas
- **fs:** se usa para exportar el canvas a una imagen
- **moment:** la hemos usado para convertir un objeto fecha a un string.
- **isolation-forest:** librería usada para utilizar un Isolation Forest ya implementado. Se ha utilizado en el algoritmo 1.
- **ml-isolation-forest:** librería usada para utilizar un Isolation Forest ya implementado. Se ha utilizado en el algoritmo 2.
- **isolation-forest-visualization:** librería usada para utilizar un Isolation Forest ya implementado. Se ha utilizado en el algoritmo 3.
- **shuffle-seed:** se utiliza para mezclar los arrays.
- **lodash:** nos aporta diferentes funciones para trabajar con arrays.

Algoritmos

Los isolation forest son un tipo de random forest, estos se encargan de encontrar las anomalías presentes de los datasets de una forma eficiente. Después de crear un nuevo isolation forest y entrenarlo, a la hora de predecir anomalías, los algoritmos le asignan un valor de anomalía, que se encuentra entre 0 y 1, a los valores de nuestro dataset. Aquellos valores de dataset con un valor de anomalía igual o superior a 0.5 serán considerados anomalías.

Para ejecutar los algoritmos debemos movernos a la carpeta *"isolation_forest"* (carpeta donde se encuentran los 3 algoritmos) y podemos ejecutar sus algoritmos individualmente o todos a la vez.

De manera grupal sería ejecutando el comando:

```
npm run algoritmos
```

Después ejecutar los algoritmos cada uno nos crea un png en la carpeta resultados con el nombre de su algoritmo correspondiente. De esa forma se puede observar mejor lo que ha sucedido. Veremos una gráfica temporal donde hay dos tipos de puntos:

- Puntos azules y pequeños: son datos normales.
- Puntos rojos y grandes: datos considerados anomalías.

Para estos algoritmos el eje X representa fechas ordenadas y el eje Y su value.

Algoritmo 1

Para este algoritmo hemos utilizado la librería isolation forest.

```
npm run algoritmo1
```

Este algoritmo es el que mejores resultados nos ha dado. Normalmente suele coger los datos más distantes.

Parámetros:

- **numberOfTrees:** Cantidad de árboles que debe generar el bosque. El valor predeterminado es 100.
- **subsamplingSize:** Tamaño de las submuestras utilizadas del conjunto de datos durante la fase de entrenamiento. Ayuda a evitar problemas comunes en la detección de anomalías. El valor predeterminado es 256 o el tamaño del conjunto de datos si es menor que 256.

Algoritmo 2

Para el segundo algoritmo hemos utilizado la librería ml-isolation-forest.

```
npm run algoritmo2
```

Es el algoritmo más simple que hemos usado puesto que no se puede parametrizar las entradas de los métodos. El resultado del algoritmo es ligeramente inferior al primero aunque a veces puede llegar a ofrecer una buena solución.

Los métodos que nos ofrece son **train**(para entrenar el algoritmo) y **predict**(para calcular los valores de anomalía)

Algoritmo 3

Para el último algoritmo hemos utilizado la librería isolation-forest-visualization.

```
npm run algoritmo3
```

En los resultados nos suele detectar datos que no deberían ser considerados anomalías. Por ese motivo es mejor utilizar el algoritmo 1.

Además cuenta con un método **dataAnomalyScore()**, que calcula el valor de las anomalías y si se le pasa como parámetro un número, mostrará por pantalla todos los datos relaciones con dichos primeros número de anomalías.

```
var myForest = new IsolationForest(features, 200,  features.length);
```

Cuando creamos el algoritmo en el código podemos pasar distintos datos:

- **features:** es un array de arrays con los datos que utilizará el algoritmo para crear los distintos árboles. **Es obligatorio pasarlo**.

- **numberOfTrees:** indica la cantidad de árboles que creará el algoritmo. Dependiendo del número que indiquemos, en caso de exportar el bosque en png se nos creará un png por cada árbol.

- **sampleSize:** número de muestra. Como máximo se puede indicar la longitud de la muestra usada. Es decir, si tenemos 50 muestras podemos colocar como máximo 50. Cuanto más alto este valor, más grandes serán los árboles creados ya que hay más muestras para ir subdividiendo el árbol.

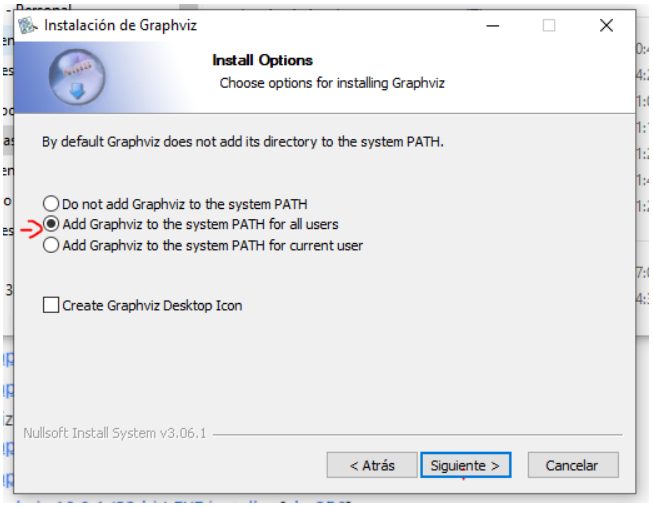
Extra: Podemos representar el isolation forest en un png, pero para ello tendremos que instalar graphviz en nuestro sistema operativo.

Lo primero será entrar en la página de graphviz, una vez dentro buscaremos el instalador de windows correspondiente para nuestro sistema.

Windows

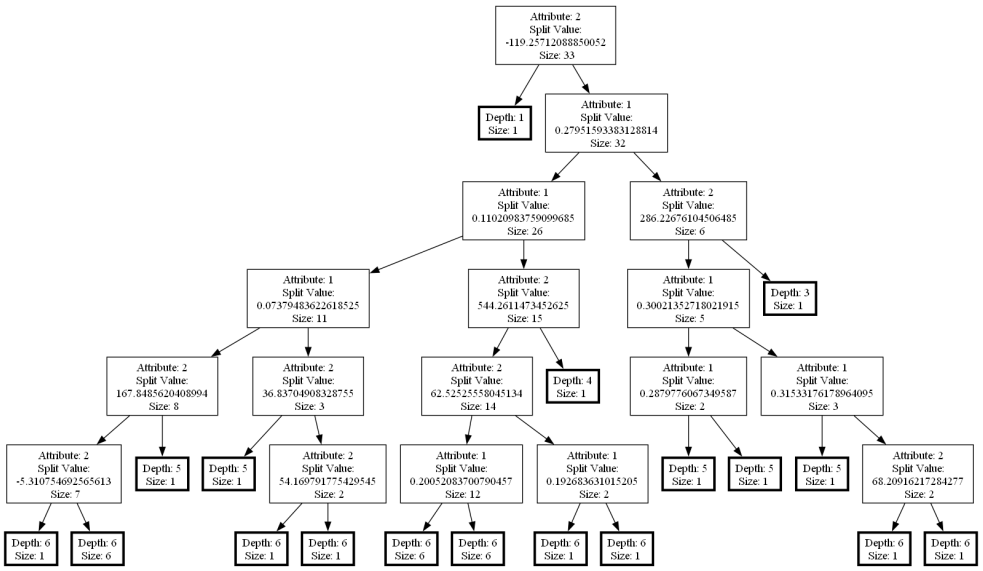
- Stable Windows install packages, built with Microsoft Visual Studio 16 2019:
  - graphviz-12.0.0
    - graphviz-12.0.0 (64-bit) ZIP archive [sha256] (contains all tools and libraries)
    - graphviz-12.0.0 (32-bit) ZIP archive [sha256] (contains all tools and libraries)
    - graphviz-12.0.0 (64-bit) EXE installer [sha256]
    - graphviz-12.0.0 (32-bit) EXE installer [sha256]
  - graphviz-11.0.0
    - graphviz-11.0.0 (32-bit) ZIP archive [sha256] (contains all tools and libraries)
    - graphviz-11.0.0 (64-bit) ZIP archive [sha256] (contains all tools and libraries)
    - graphviz-11.0.0 (32-bit) EXE installer [sha256]
    - graphviz-11.0.0 (64-bit) EXE installer [sha256]
  - graphviz-10.0.1
    - graphviz-10.0.1 (32-bit) ZIP archive [sha256] (contains all tools and libraries)
    - graphviz-10.0.1 (64-bit) ZIP archive [sha256] (contains all tools and libraries)
    - graphviz-10.0.1 (32-bit) EXE installer [sha256]
    - graphviz-10.0.1 (64-bit) EXE installer [sha256]

Durante la instalación clicamos sobre el botón siguiente/aceptar todo el rato a excepción de la siguiente ventana donde elegiremos la segunda opción:



Una vez esté todo instalado si queremos ver los resultados en un png tendremos que ejecutar el código desde la propia terminal de windows, con el mismo comando que nombramos anteriormente.

Esto genera imágenes como la siguiente:


Attribute: 2
Split Value:
-119.25712088850052
Size: 33
Depth: 1
Size: 1
Attribute: 1
Split Value:
0.27951593383128814
Size: 32
Attribute: 1
Split Value:
0.11020983759099685
Size: 26
Attribute: 2
Split Value:
286.22676104506485
Size: 6
Attribute: 1
Split Value:
0.07379483622618525
Size: 11
Attribute: 2
Split Value:
544.2611473452625
Size: 15
Attribute: 1
Split Value:
0.30021352718021915
Size: 5
Depth: 3
Size: 1
Attribute: 2
Split Value:
167.84856204089 94
Size: 8
Attribute: 2
Split Value:
36.837049083287 55
Size: 3
Attribute: 2
Split Value:
62.525255580451 34
Size: 14
Depth: 4
Size: 1
Attribute: 1
Split Value:
0.28797766673495 87
Size: 2
Attribute: 1
Split Value:
0.31533176178964095
Size: 3
Attribute: 2
Split Value:
-5.310754692565613
Size: 7
Depth: 5
Size: 1
Depth: 5
Size: 1
Attribute: 2
Split Value:
54.169791775429545
Size: 2
Attribute: 1
Split Value:
0.20052083700790457
Size: 12
Attribute: 1
Split Value:
0.19268363101520 5
Size: 2
Depth: 5
Size: 1
Depth: 5
Size: 1
Depth: 5
Size: 1
Attribute: 2
Split Value:
68.20916217284277
Size: 2
Depth: 6
Size: 1
Depth: 6
Size: 6
Depth: 6
Size: 1
Depth: 6
Size: 1
Depth: 6
Size: 6
Depth: 6
Size: 6
Depth: 6
Size: 1
Depth: 6
Size: 1
Depth: 6
Size: 1
Depth: 6
Size: 1

Librería interesante

Hemos encontrado la librería scikit-learn-ts. Es una de las librerías más usadas para machine learning en Python, pero alguien se ha dedicado a que se pueda utilizar también en ts.

Para usarla necesitas tener instalado:

- python >= 3.7
- node >=14

La librería tiene muchos algoritmos de machine learning ya implementados. Entre ellos el Isolation Forest y permite mucha parametrización. Consideramos conveniente mirar la librería y probarla por si da mejores resultados (actualmente no la hemos probado).

En la web oficial explica que está en desarrollo por eso recomienda mejor no utilizarlo en producción todavía.